Titre: Title:	Algorithme d'apprentissage du quantron basé sur Spikeprop
Auteur: Author:	Jérémie Villeneuve
Date:	2015
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Villeneuve, J. (2015). Algorithme d'apprentissage du quantron basé sur Spikeprop [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1959/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/1959/
Directeurs de recherche: Advisors:	Richard Labib
Programme: Program:	Maîtrise en mathématiques appliquées

UNIVERSITÉ DE MONTRÉAL

ALGORITHME D'APPRENTISSAGE DU QUANTRON BASÉ SUR SPIKEPROP

JÉRÉMIE VILLENEUVE
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
DÉCEMBRE 2015

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

ALGORITHME D'APPRENTISSAGE DU QUANTRON BASÉ SUR SPIKEPROP

présenté par : VILLENEUVE Jérémie
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. PARTOVI NIA Vahid, Doctorat, président
M. LABIB Richard, Ph. D., membre et directeur de recherche
M. KADOURY Samuel, Ph. D., membre

DÉDICACE

À ceux qui se passionnent pour leur métier et qui ont le coeur à l'ouvrage,
À ceux qui cherchent constamment à se dépasser tout en restant humbles,
À ceux qui savent reconnaître l'essentiel :
Vous êtes ma source d'inspiration.

REMERCIEMENTS

J'aimerais d'abord remercier mon directeur de recherche, Richard Labib, qui a toujours été disponible pour discuter et répondre à mes questions. La grande liberté qu'il m'a laissée quant à la direction du projet m'a permis d'expérimenter véritablement la nature de la recherche scientifique. De plus, le stage en industrie auquel il a contribué m'a donné la chance d'appliquer mes connaissances dans un contexte nouveau tout en agrandissant mon réseau de contacts. Je considère avoir beaucoup appris de ces expériences, autant sur les plans personnel que professionnel. Je remercie aussi Richard pour avoir bâti un cours littéralement magistral sur l'histoire des mathématiques. Il s'avère extrêmement intéressant et édifiant de prendre conscience de la façon dont les connaissances dans ce domaine ont évolué d'hier à aujourd'hui. Comme l'a affirmé Feynman : « On ne peut plus expliquer le monde, faire ressentir sa beauté à ceux qui n'ont aucune connaissance profonde des mathématiques. »

Merci à Simon de Montigny d'avoir pris le temps de partager à quelques reprises ses connaissances et ses réflexions sur l'apprentissage du quantron, sujet sur lequel il a travaillé pendant de nombreuses années. Merci également à Viviane, Étienne et Thomas pour les discussions à propos de vulgarisation scientifique, d'enseignement et d'avenir. Chacun de vous aurez contribué à éclairer un peu plus mon chemin. Merci à Sophie et à Benoît pour le support technique ! Merci enfin à mes collègues de bureau (Simon, Louis-Pierre et Fatima) pour leur éternelle bonne humeur.

Je tiens bien sûr à exprimer ma reconnaissance à mes parents qui m'ont constamment soutenu tout au long de mon cheminement académique. À leurs yeux, l'éducation a toujours été une priorité et si je dépose aujourd'hui ce mémoire, c'est en bonne partie grâce à eux.

Finalement, l'adage dit que derrière chaque grand homme se cache une femme. On ne peut certainement pas affirmer que je sois un grand homme, mais il existe néanmoins une femme qui, loin de se cacher, a été à mes côtés à chaque jour de ce parcours. En m'encourageant dans les moments creux, en partageant ma joie lors des petites victoires et en me ramenant sur le plancher des vaches lorsque ma tête flottait dans les nuages, Nathalie m'a insufflé persévérance et détermination afin que je me rende à bon port. Même s'il n'est pas directement visible, il m'apparaît donc essentiel de souligner son apport à ce travail. Ainsi, je la remercie très sincèrement pour son support et, surtout, pour son amour.

RÉSUMÉ

Le quantron est un modèle de neurone artificiel basé sur la modélisation de la diffusion des neurotransmetteurs dans la fente synaptique. Son potentiel en reconnaissance de formes a été maintes fois démontré sur des tâches de classification à frontières hautement non linéaires. Actuellement, l'exploitation de ce potentiel est restreint par l'absence d'un algorithme d'apprentissage efficace pour entraîner le quantron dans sa formulation originale.

Deux obstacles majeurs entravent l'entraînement de ce neurone. D'abord, sa sortie analogique comporte des discontinuités et ne peut être ramenée à une expression analytique et dérivable faisant intervenir les paramètres du modèle. D'autre part, à cause du formalisme de seuil d'activation inspiré du neurone biologique, il arrive que le quantron ne soit pas suffisamment excité pour transmettre l'information. La façon de gérer de tels neurones silencieux dans un contexte d'apprentissage requiert le développement de nouveaux principes. Ainsi, l'objectif principal de ce mémoire est de concevoir un algorithme qui puisse entraîner des réseaux de quantrons en proposant divers moyens pour contourner les difficultés décrites.

Le quantron partage certaines caractéristiques (sommation spatio-temporelle de potentiels postsynaptiques et mécanisme de seuil d'activation) avec la toute dernière génération de neurones artificiels, à savoir les neurones à impulsions. Ces similarités ont motivé une revue des algorithmes d'apprentissage développés pour ceux-ci ayant mené à l'identification du modèle de neurone SpikeProp (un dérivé du *Spike Response Model*) ressemblant particulièrement au quantron. L'algorithme SpikeProp associé, qui procède par rétropropagation de l'erreur et par linéarisation du potentiel de membrane autour de l'instant où le seuil est atteint, a ainsi pu être adapté au quantron avec succès. L'approximation sous-jacente à SpikeProp permet de franchir le premier obstacle associé à la non-dérivabilité de la sortie du quantron en fonction des paramètres.

Concernant le phénomène des neurones silencieux, bien qu'il survienne aussi pour les neurones impulsionnels, aucune méthode de gestion systématique et rigoureuse n'a été développée. En conséquence, on propose ici des heuristiques mesurant l'effet de chaque type de paramètre (poids synaptique, délai synaptique et demi-largeur des potentiels postsynaptiques) sur l'état d'activation du quantron dans le but de définir symboliquement certaines dérivées apparaissant dans le formalisme de rétropropagation de l'erreur. Celles-ci se fondent à la fois sur des approximations étudiées dans des travaux antérieurs, sur des résultats démontrés dans ce mémoire et sur des raisonnements intuitifs.

Pour chaque type de paramètre, cinq heuristiques sont ainsi proposées et un processus de

sélection en deux étapes est mis en place pour retenir celles qui s'avèrent optimales. Six tâches de classification binaire servent d'assise pour la comparaison des performances. Elles consistent à classer les pixels d'images de caractères alphabétiques préalablement générées par des quantrons individuels en utilisant des valeurs cibles pour la sortie du neurone et pour son état d'activation. En considérant la combinaison optimale d'heuristiques identifiée, le taux moyen de classification correcte sur les six tâches dépasse les 98 %.

L'algorithme élaboré peut aussi être appliqué en utilisant des états d'activation cibles exclusivement : les problèmes précédents ont donc été repris de cette manière. On conclut des résultats observés qu'il faut un réseau 2-10-1 pour retrouver des taux moyens de classification comparables à ceux obtenus lorsque les sorties analogiques cibles sont disponibles. Afin de poursuivre l'analyse dans ce contexte, six nouveaux problèmes de classification aux surfaces de décision variées — incluant le OU exclusif — sont introduits. La méthode est alors appliquée avec des réseaux comportant de 0 (quantron seul) à 10 unités cachées. Tandis que peu de problèmes sont résolus avec des architectures à moins de deux neurones cachés, la proportion augmente régulièrement avec la taille de la couche cachée, signe de la capacité de l'algorithme à exploiter la puissance de calcul fournie par les unités additionnelles. L'observation d'une baisse de performance des réseaux 2-1-1 par rapport aux quantrons seuls est expliquée par le fait qu'un quantron à une entrée agit comme une porte à seuil (*threshold gate*) ce qui augmente généralement la complexité de la tâche à effectuer par le neurone caché de ces réseaux. Sur cinq des six problèmes, des taux moyens de classification supérieurs à 95 % sont atteints avec des réseaux 2-10-1.

L'algorithme proposé possède plusieurs avantages, le premier étant qu'il supporte les mécanismes originaux du quantron. Effectivement, les modifications apportées à la fonction d'activation et à la sortie analogique ne modifient pas le comportement et le réalisme biologique du quantron. Comme déjà discuté, la capacité de la procédure à tirer profit de la puissance de neurones cachés supplémentaires se révèle être un autre point positif. Un dernier avantage de la méthode est la simplicité et la rapidité d'évaluation des règles d'apprentissage, puisqu'elles se basent sur les caractéristiques de la fonction d'activation en un instant seulement. L'envers de la médaille est que l'effet des paramètres sur la forme de cette dernière risque d'être mal évalué par le fait même de cette simplicité. Considérer un instant précis signifie également considérer un nombre réduit de potentiels postsynaptiques et il est possible que cela soit la cause de plusieurs essais divergents pour lesquels l'algorithme donne une trop grande magnitude au poids d'une entrée tout en étouffant les autres. Un dernier inconvénient apparaît dans une situation bien spécifique où l'apprentissage stagne momentanément lorsque toute une couche de neurones reste simultanément silencieuse suite à un stimulus. De telles circonstances demeurent toutefois assez rares.

Les bonnes performances livrées par ce nouvel algorithme encouragent à l'investiguer davantage et à proposer des ajouts et des modifications pour en accroître l'efficacité. Les travaux futurs consisteraient notamment à implémenter des règles d'apprentissage tenant compte de l'aspect global de la fonction d'activation (par opposition à ses propriétés en un temps précis), à élaborer une méthode d'initialisation des paramètres d'un réseau qui minimiserait la proportion d'essais divergents, et d'appliquer la méthode à des problèmes multiclasses en recourant à de multiples neurones de sortie.

ABSTRACT

The quantron is an advanced artificial neuron based on the mathematical modelling of neurotransmitter diffusion in the synaptic cleft. Its potential in pattern recognition was established on classification tasks showing highly nonlinear decision boundaries. Currently, this potential can not be fully exploited since no learning algorithm has been proposed to train the quantron in its exact form.

Two main aspects hinders the training of the quantron. On one hand, its analog output suffers from discontinuities and does not have an analytic and differentiable expression with respect to model parameters. On the other hand, due to the underlying threshold firing mechanism inspired from biological neurons, the quantron can block information transmission if not sufficiently excited by the input pattern. The question of how to manage such silent neurons in a learning context is complex and new principles needs to be developed in this regard. Therefore, the main objective of this work is to design an algorithm for training multilayer quantrons implementing various means to overcome the aforementioned impediments.

The quantron shares some characteristics (the spatiotemporal summation of postsynaptic potentials and the threshold firing mechanism) with the latest generation of artificial neurons, namely the spiking neurons. These similarities motivated a literature review of spiking neuron learning algorithms which led to the identification of the SpikeProp neuron (a simplified form of the Spike Response Model) as a model highly similar to the quantron. This allowed the associated SpikeProp algorithm — implementing error backpropagation and membrane potential linearization around the firing time — to be successfully adapted to the quantron to overcome the problem of the non-differentiability of the analog output with respect to parameters.

Even though spiking neurons are also subject to block information, no systematic, rigorous method was developed to deal with such silent units. Consequently, this work proposes heuristics assessing the effect of each type of parameter (synaptic weight, synaptic delay and postsynaptic potential half-width) on the activation state of the quantron. This way, activation state derivatives appearing in the error backpropagation paradigm are defined unambiguously. These heuristics are based on previous work approximations, results proven in this dissertation and intuitive reasoning.

For each parameter type, five heuristics are built and a two-step selection process is set up to retain the optimal ones. Six binary classification tasks are used to assess performance. Each consists in classifying the pixels of a binary image representing an alphabetic character

previously generated by single quantrons using target values for the neuron's analog output and activation state. When run using the optimal set of heuristics, the method yields a classification rate exceeding 98 % when averaged over all six tasks.

Since the algorithm can be applied using target activation states only, the previous problems were reconsidered in this fashion. Results show that 2-10-1 networks are necessary to achieve comparable mean classification rates than those obtained when target analog output values are available. To investigate further in this direction, six new classification problems having varied decision boundaries — including the well-known XOR problem — are introduced. Architectures with hidden layer size ranging from 0 (single quantron) to 10 hidden neurons are trained. While very few problems are perfectly solved with networks having less than two hidden units, this proportion steadily increases with hidden layer size. This behavior is evidence of the ability of the algorithm to exploit extra computation power made available by additional units. The performance drop observed for 2-1-1 networks is explained by the fact that a single-input quantron acts as a threshold gate and that this generally increases the complexity of the problem to be solved by the hidden neuron. On five problems out of six, mean classification rates exceeding 95 % are obtained with 2-10-1 architectures.

The new method exhibits many advantages, the first being that it works with the original quantron mechanism. Indeed, even though modifications were made to the activation function and to the analog output, neither of them alters its behavior or its biological realism. As was mentioned previously, another positive aspect is that it is able to take advantage of additional hidden neurons to solve complex problems. Also, the evaluation of learning rules is simple and quick since it relies on properties of the activation function at a single time. The inherent drawback is that it may oversimplify the effect of a parameter modification on the shape of the activation function. Moreover, considering a single time often means considering a small number of postsynaptic potentials: that is possibly the cause for many divergent runs where the algorithm gave too much importance to a single input (by increasing its weight) and neglecting or suppressing the others. Another drawback appears in a very special case where learning momentarily stops if a whole layer of neurons remains silent following an input pattern. Fortunately, such a condition is hardly ever met.

The good performance delivered by this novel algorithm suggests that it is worth investigating further in this direction to devise extensions increasing its efficacy. Future work would include creating improved heuristics taking into account the global shape of the activation function instead of relying on a single point, designing a better parameter initialization procedure minimizing the proportion of divergent runs, and applying the method to multiclass problems using multiple output neurons.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

BP	Rétropropagation de l'erreur (*Error BackPropagation*)
LIF	Neurone *Integrate-and-Fire* avec fuite
LSM	Machine à état liquide (*Liquid State Machine*)
LTD	Dépression à long terme (*Long Term Depression*)
LTP	Potentialisation à long terme (*Long Term Potentiation*)
MLP	Réseau de perceptrons multicouche (*Multilayer perceptron*)
MLQ	Réseau de quantrons multicouche (*Multilayer quantron*)
NT	Neurotransmetteurs
PA	Potentiel d'action
PPS	Potentiel postsynaptique
ReSuMe	Remote Supervised Method
RNA	Réseau de neurones artificiels
RNI	Réseau de neurones à impulsions
SRM	*Spike Response Model*
STDP	*Spike-Timing-Dependant Plasticity*

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

1.1 Les machines peuvent-elle penser ?

La volonté de construire des machines reproduisant les caractéristiques humaines ne date pas d'hier. Au deuxième siècle av. J.-C., la Grèce antique est déjà le berceau du premier calculateur analogique : la machine d'Anticythère. Ses engrenages en bronze savamment interconnectés servaient alors à prédire les éclipses lunaires et solaires ainsi que les phases de la Lune (Freeth, 2009). Bien que son fonctionnement soit purement mécanique, les connaissances de l'époque en astronomie y avaient été sciemment encryptées à travers le choix du nombre de dents de chaque engrenage et de leur disposition. Ainsi, la machine pouvait réaliser une tâche de prédiction pouvant être qualifiée « d'intelligente » dans la mesure où seul un humain pouvait y parvenir jusqu'alors.

Depuis le siècle dernier, l'invention, la démocratisation et l'augmentation soutenue de la puissance de l'ordinateur ont permis de développer des automates de plus en plus crédibles quant à leur habiletés. En 1997, le superordinateur *Deep Blue* d'IBM a réalisé un exploit sans précédent en remportant une série de parties d'échecs contre le champion mondial de l'époque, Garry Kasparov (Hsu, 2002). En 2011, *Watson*, un autre superordinateur d'IBM, a vaincu deux anciens gagnants du jeu télévisé *Jeopardy* en répondant en temps réel aux questions qui lui étaient posées en langage naturel [1]. Enfin, certains logiciels, comme *Cleverbot* ou *Eugene Goostman*, imitent suffisamment bien la conversation écrite d'un être humain pour réussir à tromper une proportion significative de juges devant déterminer si leur interlocuteur virtuel est de nature humaine ou non [2].

Toutes les tâches citées ci-dessus font appel à différentes compétences intellectuelles complexes : reconnaître un langage, l'interpréter, puis user de ses connaissances et de son expérience afin de produire une réponse optimale. Pourtant, tous ces automates sont régis par des algorithmes et des procédures qu'ils sont contraints de suivre à la lettre et qui proscrivent toute forme de spontanéité. Pour reprendre la célebre question que s'est posée Alan Turing au milieu du siècle dernier, c'est donc dire que ces machines ne pensent pas réellement, pour

1. Voir l'article du *New York Times* paru le 16 février 2011 : `http://www.nytimes.com/2011/02/17/science/17jeopardy-watson.html?pagewanted=all&_r=0`.

2. Cela a d'ailleurs donné lieu à certaines controverses à savoir si elles ont réussi ou non à passer le célèbre test de Turing qui vise à mesurer la capacité d'une machine à agir comme un humain. Voir l'article du journal *The Guardian* paru le 9 juin 2014 : `http://www.theguardian.com/technology/2014/jun/08/super-computer-simulates-13-year-old-boy-passes-turing-test`.

autant que l'on considère que suivre un ensemble de règles dictées ne caractérise pas l'acte de pensée. En fait, si ces machines dégagent une si forte impression d'intelligence, c'est grâce aux méthodes sophistiquées d'analyse de données et de reconnaissance de formes qu'elles déploient. Sans celles-ci, *Watson* ne pourrait ni comprendre la question qui lui est posée, ni identifier la réponse la plus plausible, ni décider s'il est avantageux de répondre en fonction de son niveau de confiance en cette dernière.

Les réseaux de neurones artificiels (RNA) forment une branche particulière de ces méthodes dites d'apprentissage machine. La particularité des RNA réside dans leur fonctionnement qui s'inspire de la structure biologique des neurones du cerveau. Au coeur de cette analogie trône la neuroplasticité, cette propriété qui permet aux neurones de s'adapter à leur environnement en modifiant leur organisation et leurs interactions. Ce mémoire porte ainsi sur l'apprentissage du quantron, un modèle récent de neurone artificiel ayant montré un potentiel certain en reconnaissance de formes. La section qui suit aborde rapidement les RNA d'un point de vue général avec l'intention d'introduire le quantron par la suite.

1.2 Réseaux de neurones artificiels

Les réseaux de neurones artificiels sont des assemblages d'unités de calcul non linéaires interconnectées les unes aux autres. Lorsqu'un vecteur d'entrées est présenté au réseau, chaque unité de base traite et propage l'information jusqu'à ce qu'un vecteur de réponses soit produit en sortie. On dénote un réseau acyclique par le nombre d'éléments dans chacune de ses couches : la Figure 1.1 représente ainsi un réseau 2-5-1 à deux entrées, cinq neurones cachés et un neurone de sortie. Les RNA sont notamment utilisés en régression, en classification et en *clustering*.

La structure de réseau rappelle celle des neurones biologiques qui entretiennent un grand nombre de connexions avec les autres neurones de leur environnement. Les liens entre ces unités sont modélisés par des paramètres qui affectent le comportement du réseau. Pour être utile, un RNA doit posséder un *algorithme d'apprentissage*, c'est-à-dire un ensemble de règles encadrant les ajustements à apporter aux paramètres afin que le réseau accomplisse le mieux possible une tâche donnée. Par exemple, en classification, le réseau est stimulé par des exemples tirés d'un ensemble d'entraînement. Une classe étant attribuée préalablement à chaque exemple, l'algorithme d'apprentissage fait en sorte que si le réseau commet une erreur en classant mal un exemple qui lui est présenté, ses paramètres sont modifiés afin que cette erreur s'annule éventuellement. Cette situation est un cas typique d'*apprentissage supervisé*, car une réponse cible est attendue suite à la présentation de chaque exemple. Si la procédure d'apprentissage réussit, l'information contenue dans l'ensemble d'entraînement est assimilée

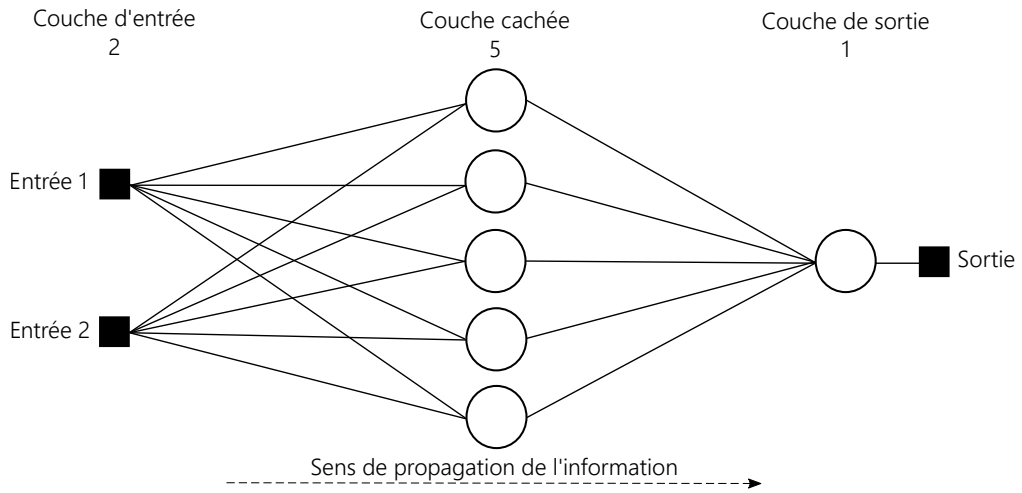


Figure 1.1 Schéma d'un réseau acyclique de neurones artificiels 2-5-1

et encodée par les valeurs spécifiques prises par les paramètres du réseau, exactement comme l'expérience et les connaissances des Anciens ont été jadis inculquées à la machine d'Anticythère à travers les fins détails de son mécanisme. Le réseau peut alors être utilisé avec de nouvelles données à des fins de *généralisation*.

Il existe une grande variété de modèles de neurones articifiels selon leurs niveaux de réalisme biologique et de complexité mathématique. Ils peuvent être catégorisés en se basant sur la façon dont l'information est encodée et transmise. Les neurones de première génération tels que le neurone de McCulloch-Pitts (McCulloch et Pitts, 1943) traitent de l'information discrète. Typiquement, ils ont deux réponses possibles correspondant aux états binaires 0 et 1. Les neurones de seconde génération comme le perceptron (Rosenblatt, 1958) transigent pour leur part avec des valeurs continues, ce qui leur permet de travailler avec des signaux analogiques. Enfin, les neurones à impulsions composent la troisième génération de neurones pour lesquels les entrées et sorties sont des trains d'impulsions (Maass, 1997). Il sera vu à la Section 1.3 que le quantron se positionne quelque part entre les deux dernières générations : bien qu'il reçoive et produise des valeurs continues, son mécanisme interne de fonction d'activation et de seuil est partagé avec les neurones à impulsions.

Afin de bien comprendre le fonctionnement d'un neurone artificiel et particulièrement celui du quantron, on présente succintement le neurone biologique dont l'anatomie est schématisée à la Figure 1.2. D'abord, l'information provenant des neurones voisins est reçue aux dendrites et au corps cellulaire sous forme de signaux électriques appelés potentiels postsynaptiques (PPS). Une fois générés, ceux-ci se propagent et se rejoignent au soma. La transmission du

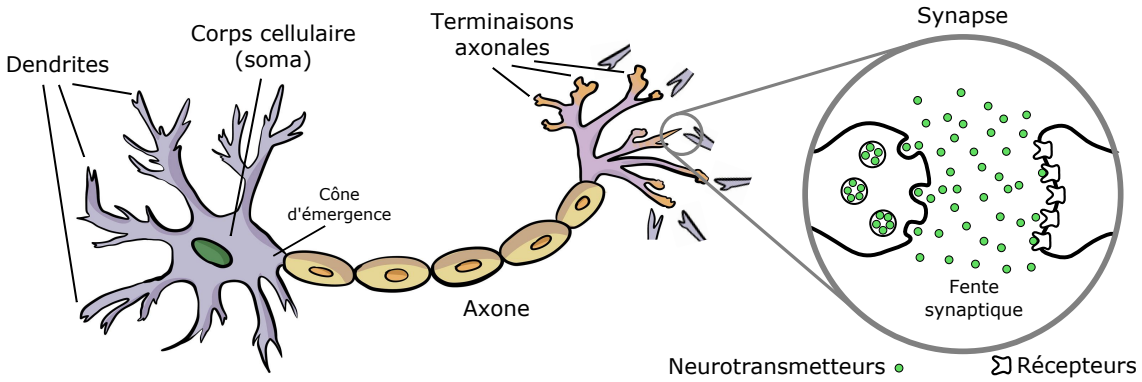


Figure 1.2 Schéma d'un neurone biologique

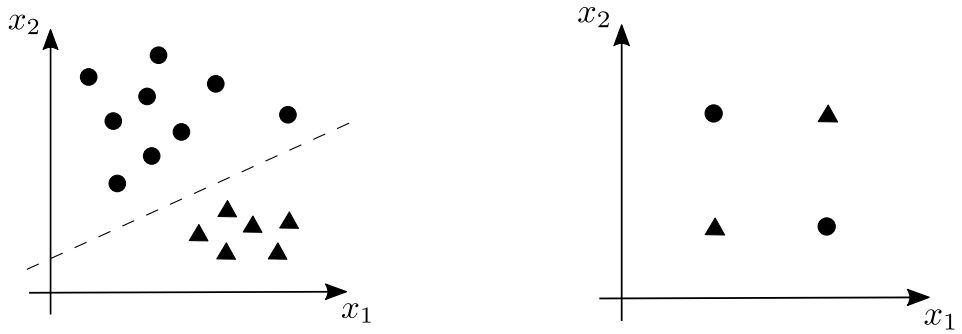
signal est possible grâce à l'ouverture des canaux ioniques de la membrane (Na^{2+} et K^{+} principalement) activés par différence de potentiel. En s'additionnant tant spatialement que temporellement au soma, les PPS y perturbent le potentiel de membrane, soit la différence de potentiel entre l'intérieur et l'extérieur du neurone. Un PPS peut ainsi être excitateur et faire augmenter ce potentiel, ou être inhibiteur et le faire décroître. Si, à un instant donné, le potentiel au cône d'émergence atteint un seuil d'activation, un potentiel d'action (PA) est généré. Un PA diffère d'un PPS principalement par son amplitude plus élevée et par sa forme distincte. Le PA est propagé et relayé le long de l'axone, avant de rejoindre les terminaisons axonales contenant des vésicules de neurotransmetteurs (NT). L'effet d'un PA est d'y activer les canaux calciques pour provoquer l'exocytose, à savoir le relâchement des NT dans la fente synaptique (Wickens, 2009). Par diffusion, une certaine proportion des NT traverse la synapse et rejoint les dendrites d'un autre neurone. Les NT peuvent alors se lier aux récepteurs qui s'y trouvent et déclencher la génération de PPS dans le neurone postsynaptique, perpétuant ainsi le cycle de transmission de l'information [3].

Toutes les générations de neurones partagent le concept d'additionner leurs entrées en les pondérant par des poids mesurant l'efficacité des synapses, afin de simuler l'accumulation des PPS au soma. La différence se situe au niveau de l'interprétation de la sortie du neurone. Alors que le modèle de McCulloch-Pitts transmet si oui ou non au moins un PA est généré, la sortie d'un perceptron peut être interprétée comme étant le taux moyen de génération de PA (Maass, 1997). Celle du quantron s'interprète similairement et symbolise, quant à elle,

3. Le mécanisme de transmission de l'information par NT est caractéristique d'une synapse chimique. Il existe d'autres types de synapses, notamment la synapse électrique dans laquelle le signal est transmis directement grâce à des jonctions communicantes. Une synapse électrique possède l'avantage d'être plus rapide, ce qui la rend particulièrement adéquate pour les réflexes primitifs de sursaut et de fuite (Dermietzel et Spray, 2013).

l'inverse du taux d'activité du neurone. Puisque la sortie des neurones à impulsions est formée de PA individuels tout comme pour le neurone biologique, ces derniers s'avèrent davantage réalistes.

En contrepartie, ce réalisme complexifie grandement leur apprentissage, ce qui les désavantage par rapport au perceptron qui, lui, bénéficie de l'algorithme de rétropropagation de l'erreur (BP) pour être entraîné en réseau multicouche (MLP) (Rumelhart *et al.*, 1986). Il faut savoir qu'un perceptron seul est limité à résoudre des problèmes de classification binaire *linéairement séparables* (Haykin, 1999) dont les classes peuvent être isolées parfaitement l'une de l'autre par un hyperplan (voir Figure 1.3), ce qui réduit passablement l'ensemble des formes qu'il peut reconnaître. Par contre, un réseau avec suffisamment d'unités dans sa couche cachée possède les propriétés d'un approximateur universel au sens où il peut reproduire avec un niveau de précision arbitraire n'importe quelle fonction réelle continue sur un compact de $\mathbb{R}^M$, $M \in \mathbb{N}^*$ (Hornik *et al.*, 1989) [4]. Ainsi, avant de disposer de la BP pour entraîner les MLP, les nombreux théorèmes négatifs démontrés par Minsky et Papert pour un seul percpetron ainsi que leur conjecture stipulant que ses avantages ne resteraient pas nécessairement vrais dans une structure de réseau (Minsky et Papert, 1988) avaient jeté de sérieux doutes sur le potentiel des RNA. L'avènement de la BP a revigoré ce champ en démentant cette conjecture et les perceptrons restent largement utilisés encore aujourd'hui.



(a) Problème linéairement séparable

(b) Problème du XOR (non linéairement séparable)

Figure 1.3 Exemples de problèmes de classification binaire sur $\mathbb{R}^2$

Cela illustre bien les possibilités qu'apportent l'élaboration d'un bon algorithme d'apprentissage pour un réseau de neurones. À l'heure actuelle, le quantron dans sa formulation originale ne possède pas de tel algorithme. L'intention derrière ce mémoire étant de faire un pas de

4. Cette condition n'est pas très restrictive et est respectée par la grande majorité des fonctions considérées en pratique.

plus dans cette direction, le quantron est introduit à la section suivante.

1.3 Le quantron

Le quantron est un modèle original de neurone artificiel basé sur la transmission de l'information par la diffusion de neurotransmetteurs dans la synapse biologique (Labib, 1999). Le quantron possède deux avantages qui motivent son étude. D'une part, il présente un réalisme biologique accru par rapport au perceptron. D'autre part, un seul quantron à six paramètres peut résoudre le problème du OU-exclusif (XOR, voir Figure 1.3b) non linéairement séparable alors qu'un MLP à sept paramètres est requis pour cette même tâche (Labib, 1999). Le quantron détient donc une certaine supériorité en terme de potentiel en reconnaissance de formes qu'il serait bien sûr souhaitable d'exploiter davantage.

1.3.1 Formalisme

Un quantron à $M \in \mathbb{N}^*$ entrées à $N \in \mathbb{N}^*$ potentiels postsynaptiques est schématisé à la Figure 1.4.

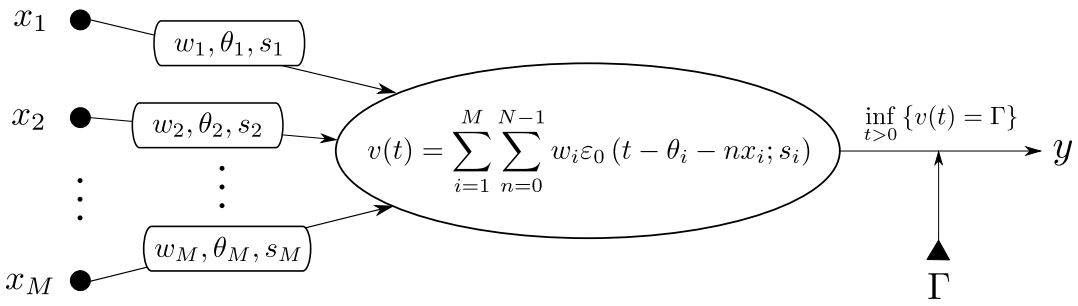


Figure 1.4 Schéma d'un quantron à M entrées à N PPS chacune

D'abord, un vecteur d'entrées réelles $\mathbf{x} = (x_1, \ldots, x_M) \in \mathbb{R}_+^M$ représente l'arrivée d'un train de PPS en provenance des synapses établies avec les neurones de la couche précédente. Plus précisément, la valeur de x_i symbolise l'inverse de leur fréquence d'arrivée. À chaque entrée est associé un triplet de paramètres (w_i, θ_i, s_i) caractérisant la synapse correspondante. Le poids synaptique $w_i \in \mathbb{R}$ module la force de la synapse, le délai synaptique $\theta_i \in \mathbb{R}_+$ représente son délai de transmission et la demi-largeur $s_i \in \mathbb{R}_+^*$ caractérise l'étendue temporelle des PPS générés aux dendrites qui dépend de la durée des liaisons entre les NT et récepteurs. Le temps d'arrivée t_{in} du n-ième PPS en provenance de l'entrée i est donné par

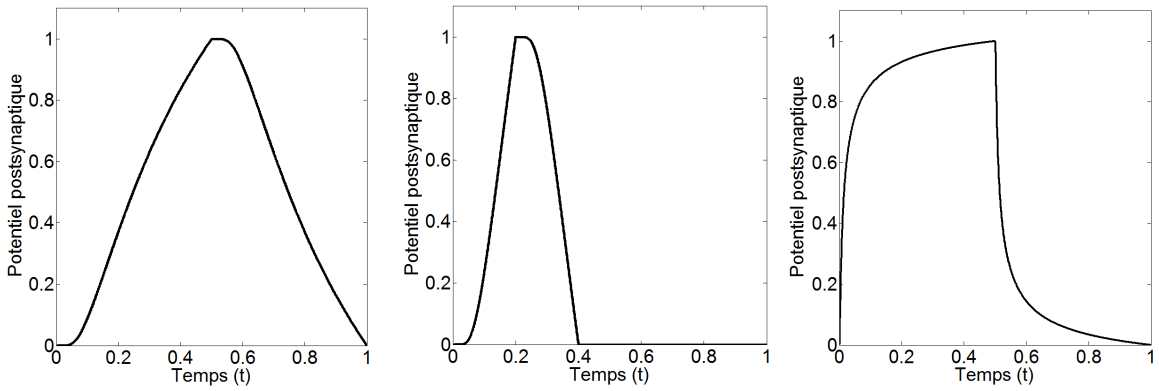
$$t_{in} = \theta_i + n x_i, \qquad i = 1, \ldots, M \qquad n = 0, \ldots, N-1. \tag{1.1}$$

Un PPS possède une phase de dépolarisation où le potentiel électrique de la membrane

cellulaire augmente suite à l'arrivée des NT, suivie d'une période de repolarisation durant laquelle le potentiel revient à sa valeur de repos [5]. La forme précise d'un PPS est représentée par le noyau

$$\varepsilon_0(t;s) = \begin{cases} cQ(\frac{\ln a}{\sqrt{t}}) & \text{si } 0 \leq t < s \\ c\left[Q(\frac{\ln a}{\sqrt{s}}) - Q(\frac{\ln a}{\sqrt{t-s}})\right] & \text{si } s \leq t < 2s \\ 0 & \text{sinon} \end{cases}, \tag{1.2}$$

où $Q(z)$ est la probabilité qu'une variable aléatoire normale centrée réduite soit supérieure à z et $c \in \mathbb{R}_+^*$ est une constante de normalisation. Cette expression particulière a été déduite à partir d'une modélisation stochastique de la diffusion des NT dans une fente synaptique unidimensionnelle de longueur $a \in \mathbb{R}_+^*$ par un mouvement brownien géométrique (Labib, 1999). La notation $\varepsilon_0(t;s)$ insiste sur le fait que le paramètre s influence la forme des PPS en modifiant leur étendue temporelle tandis que l'indice 0 indique qu'il s'agit de la forme originale des PPS associée au quantron. Afin de manipuler des PPS d'amplitude unitaire ($\max \varepsilon_0(t;s) = 1$), on fixe $c = Q(\frac{\ln a}{\sqrt{s}})^{-1}$. La forme d'un tel PPS est illustrée à la Figure 1.5 pour quelques valeurs de a et de s. On y remarque que les formes des phases de dépolarisation et de repolarisation sont symétriques : l'une peut être retrouvée de l'autre par une symétrie par rapport à l'axe horizontal $\varepsilon_0(t;s) = 1/2$.



(a) $s = 0{,}5\,;\ a = 2$ (b) $s = 0{,}2\,;\ a = 2$ (c) $s = 0{,}5\,;\ a = 1{,}1$

Figure 1.5 Forme $\varepsilon_0(t;s)$ d'un PPS du quantron pour différentes valeurs de s et de a

À la manière du corps cellulaire qui accumule les signaux en provenance des dendrites, le quantron additionne tous les PPS modulés par le poids synaptique associé à l'entrée corres-

5. En fait, cette forme est typique d'un PPS excitateur. Il existe également des PPS inhibiteurs pour lesquels une phase d'hyperpolarisation caractérisée par une baisse de potentiel survient avant que le potentiel ne remonte à sa valeur de repos. Dans le modèle du quantron, la nature excitatrice ou inhibitrice du PPS est déterminée par le signe du poids synaptique qui lui est affecté, voir équation (1.3).

pondante. Ainsi, la contribution du train de PPS $\nu_i(t)$ associé à l'entrée i s'exprime par

$$\nu_i(t) = \sum_{n=0}^{N-1} w_i \varepsilon_0(t - t_{in}; s_i) = \sum_{n=0}^{N-1} w_i \varepsilon_0(t - \theta_i - n x_i; s_i). \tag{1.3}$$

Le premier PPS du train ($n = 0$) débute à l'instant $t = \theta_i$, le deuxième PPS ($n = 1$) commence à l'instant $t = \theta_i + x_i$ et il en va ainsi jusqu'au N-ième et dernier PPS qui survient à l'instant $t = \theta_i + (N-1)x_i$. Ainsi, le train est constitué de N PPS de largeur $2s_i$ régulièrement espacés par des intervalles de temps de durée x_i.

Le résultat de la somme sur les entrées, noté $v(t)$, est équivalent au potentiel de membrane et est appelé fonction d'activation.

$$v(t) = \sum_{i=1}^{M} \nu_i(t) = \sum_{i=1}^{M} \sum_{n=0}^{N-1} w_i \varepsilon_0(t - \theta_i - n x_i; s_i) \tag{1.4}$$

De manière analogue au neurone biologique, la sortie du quantron est déterminée par le maximum global atteint par $v(t)$. Si $v(t)$ atteint le seuil d'activation $\Gamma \in \mathbb{R}_+$, alors le quantron transmet l'information et sa sortie prend la valeur du temps où la fonction d'activation atteint le seuil pour la première fois. Dans le cas contraire, la transmission d'information est bloquée et le quantron reste *silencieux*. On associe une sortie infinie à un tel état. Sous ces conditions, la sortie du quantron s'exprime de manière concise grâce au concept d'infimum sous la convention $\inf \emptyset = \infty$.

$$y = \inf \{t > 0 \mid v(t) = \Gamma\} \tag{1.5}$$

La sortie peut être utilisée comme entrée d'un quantron subséquent et il est alors possible de former des réseaux de quantrons multicouches (MLQ).
Enfin, il est utile de définir l'état d'activation z du quantron afin de savoir s'il transmet l'information ou non.

$$z = \begin{cases} 0 & \text{si } y = \infty \\ 1 & \text{autrement} \end{cases} \tag{1.6}$$

Comme proposé par de Montigny (2014), on peut alors écrire (1.4) sous la forme

$$v(t) = \sum_{i=1}^{M} \sum_{n=0}^{N-1} z_i w_i \varepsilon_0(t - \theta_i - n x_i; s_i), \tag{1.7}$$

ce qui élimine toute ambiguïté possible dans le cas où au moins une entrée est inactive. En

effet, même si l'interprétation du terme d'indice $n = 0$ devient nébuleuse lorsque $x_i = \infty$, le facteur multiplicatif $z_i = 0$ assure que l'entrée i ne contribue pas à la fonction d'activation.

La Figure 1.6 illustre des exemples des deux états d'activité possibles du quantron. Ces états ont été traditionnellement utilisés pour résoudre des problèmes de classification binaire comme le XOR : le quantron doit être actif lorsque stimulé par une classe d'exemples, tandis qu'il doit rester silencieux en présence des exemples de l'autre classe.

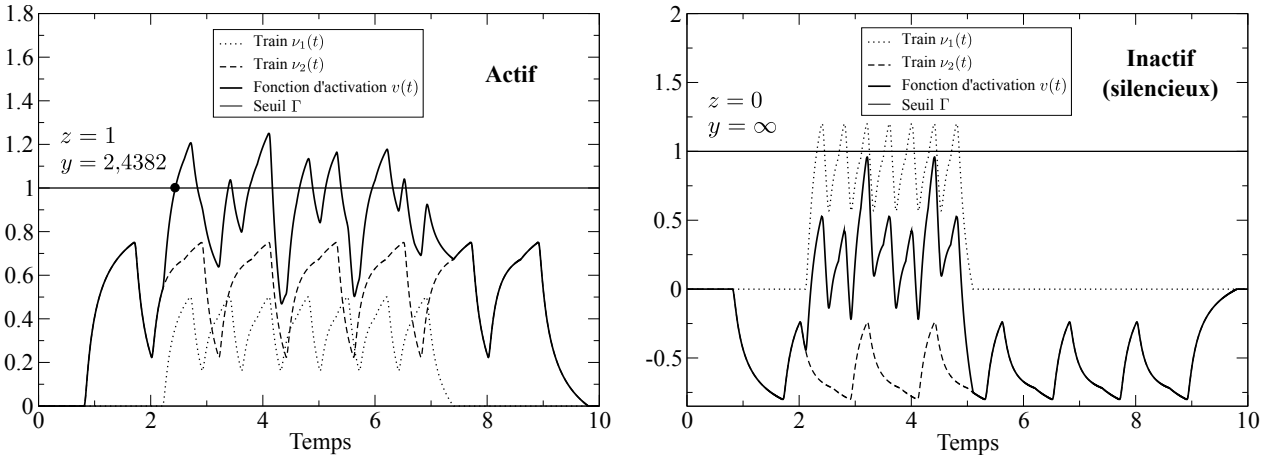


Figure 1.6 États d'activation possibles d'un quantron à deux entrées

Tout comme le perceptron, les entrées et la sortie d'un quantron sont statiques, i.e. elles ne dépendent pas du temps. Par contre, la dynamique temporelle sous-jacente introduite par $v(t)$ et Γ a pour conséquence que la sortie du quantron ne peut s'exprimer analytiquement en fonction de ses paramètres comme c'est le cas pour le perceptron. Cette particularité permet des interactions complexes entre les entrées dans lesquelles le quantron puise toute sa puissance de calcul. Toutefois, cette même complexité amène de grands défis quant à son apprentissage.

1.3.2 Travaux antérieurs

Les travaux antérieurs sur le quantron se subdivisent en deux axes principaux. Le premier s'intéresse à son potentiel en reconnaissance de formes tandis que le second se concentre sur l'apprentissage supervisé du neurone.

Potentiel du quantron

Une étude du fonctionnement interne du quantron a permis de confirmer son potentiel en générant les 26 lettres majuscules de l'alphabet latin à l'aide de MLQ à noyau triangulaire et

d'opérations logiques (Hackenbeck-Lambert, 2011). Dans l'optique d'accroître ce potentiel, des éléments de biomimétisme plus fins ont été incorporés. Ainsi, une modélisation poussée du parcours synaptique des NT a permis d'établir une forme analytique des PPS reproduisant les mesures expérimentales en neurophysiologie (L'Espérance, 2010). Également, deux phénomènes biologiques importants ont été intégrés. La dépression synaptique, une diminution de l'amplitude des PPS due à une activité élevée du neurone, et les périodes réfractaires absolues et relatives, qui empêchent et défavorisent respectivement l'émission d'un nouveau PA suite à un premier, ont ainsi été modélisées par des processus stochastiques. Ces innovations ont rendu possible la construction d'îlots circulaires par un seul quantron (Hamel, 2013).

Apprentissage supervisé du quantron

Encore à ce jour, il n'existe pas d'algorithme général qui permette, pour une banque d'exemples donnée, de retrouver un ensemble de paramètres cohérents tel qu'un MLQ reproduise les sorties associées aux entrées de l'ensemble d'entraînement. Plusieurs ont orienté leurs travaux dans cette direction.

Une approche multiéchelle de type « ondelettes » à base de paraboles tronquées a été proposée pour approcher le maximum de la fonction d'activation d'un quantron (Connolly et Labib, 2009) qui joue un rôle important dans l'atteinte du seuil. Un algorithme consistant à exciter ou à inhiber le quantron en cas de neurone silencieux et à utiliser autrement la rétropropagation avec le temps de sortie du quantron a été expérimenté sur un MLQ (Lastère, 2005). Celui-ci utilise une approximation de deuxième ordre de la fonction d'activation du quantron par série de Fourier paire afin de résoudre le problème du XOR avec un taux de réussite de 75 %. Sur les problèmes de reconnaissance d'une forme linéairement séparable et d'une forme générée par un MLQ, l'algorithme parvient à réduire l'erreur sans toutefois être capable de la rendre nulle. Un autre algorithme fondé sur la rétropropagation des temps de sortie du quantron a été appliqué à la reconnaissance de caractères (Pepga Bissou, 2007). Les états binaires de tous les pixels de l'image sont fournis en entrée au MLQ qui doit alors émettre à un temps prédéfini associé au caractère. Dans cette méthode, la réponse de chacune des entrées est approchée par une parabole afin d'obtenir des dérivées continues de l'erreur par rapport aux paramètres (w, θ, s) du quantron.

Suivant cette idée, une avenue explorée en classification est d'utiliser des polynômes quadratiques pour représenter des trains de PPS carrés. Deux algorithmes d'apprentissage, l'un par fonctions discriminantes et l'autre par descente du gradient, ont été développés dans cette lignée (de Montigny, 2007). Il arrive cependant que les coniques générées par les approxima-

tions quadratiques ne parviennent pas à reproduire fidèlement les surfaces discriminantes du quantron exact ce qui constitue un inconvénient notable.

Une étude sur les PPS de forme carrée a permis d'établir une expression pour le maximum d'un train de PPS. Un quantron a ainsi pu être entraîné par descente du gradient en jumelant des approximations analytiques *softmin* et *softmax* à des approximations par polynômes quadratiques (Labib et De Montigny, 2012) sur le problème IRIS (Fisher, 1936). Ce dernier contient 50 spécimens de fleurs de trois espèces du genre *Iris* décrits par quatre caractéristiques (longueur et largeur des pétales et des sépales) : l'objectif consiste alors à classer correctement chaque spécimen selon l'espèce à laquelle il appartient. Les résultats montrent que les performances de trois quantrons sur le problème IRIS sont comparables à celles de trois perceptrons si les deux principales caractéristiques des spécimens sont retenues.

Le problème de classification binaire a également été abordé d'un point de vue géométrique (de Montigny et Labib, 2011). En utilisant des PPS de formes carrée et rampe, le maximum de $v(t)$ est décrit analytiquement et les paramètres du quantron sont mis en correspondence avec les caractéristiques géométriques des surfaces discriminantes. Bien qu'originale, cette approche est toutefois limitée à un seul quantron à deux entrées.

Plus récemment, l'algorithme de rétropropagation lisse a été proposé pour le MLQ à noyau triangulaire (de Montigny, 2014). D'abord, une approximation par sigmoïde de l'état d'activation du neurone est intégrée comme facteur multiplicatif dans la fonction d'activation.

$$z \underset{\lambda\to\infty}{\sim} \sigma(\max v(t) - \Gamma; \lambda) = \frac{1}{1 + \exp\left[\lambda \left(\max v(t) - \Gamma\right)\right]} \tag{1.8}$$

Ensuite, la forme asymptotique suivante pour le maximum de $v(t)$

$$\max_{0\le t\le T} v(t) \underset{\lambda\to\infty}{\sim} \frac{1}{\lambda} \ln\left(\int_0^T e^{\lambda v(t)} dt\right) \tag{1.9}$$

est employée pour développer une approximation de la sortie y. Les approximations (1.8) et (1.9) s'avèrent d'autant plus précises que la valeur du paramètre d'approximation positif λ est élevée. Ces manipulations permettent d'une part d'exprimer la sortie du quantron analytiquement en fonction de ses paramètres et de ses entrées, et d'autre part d'éliminer le problème des neurones silencieux qui entravent normalement l'apprentissage du MLQ en ne transmettant aucune information aux neurones suivants. Cela résulte en un algorithme de descente du gradient entièrement constitué d'expressions analytiques, quoique relativement complexes.

De toutes les approches présentées précédemment, l'algorithme de rétropropagation lisse s'approche le plus d'une procédure d'apprentissage supervisé complète pour le quantron, seul ou en réseau. L'approche n'est toutefois pas parfaite. En effet, il arrive que la solution trouvée avec le formalisme approximatif perde ses qualités une fois injectée dans le modèle sans approximation. En théorie, cet inconvénient peut être contourné car l'approche gagne en précision plus le paramètre d'approximation λ est élevé. Cela dit, il faut alors porter une attention particulière au débordement arithmétique lors de l'implémentation.

1.4 Plan et objectifs du mémoire

L'objectif des travaux de ce mémoire est d'établir un algorithme d'apprentissage pour le MLQ en s'inspirant de méthodes déjà existantes pour d'autres types de neurones partageant certaines caractéristiques avec le quantron. Comme mentionné à la Section 1.3.1, le fait que la sortie du quantron ne possède pas d'expression analytique en fonction des paramètres constitue définitivement un obstacle important. Or, la dynamique temporelle qui en est responsable se retrouve également dans les neurones à impulsions. L'idée consiste alors dans un premier temps à identifier et à formaliser les ressemblances entre ces derniers et le quantron. Dans un second temps, il s'agit de s'inspirer des stratégies déployées par les algorithmes d'apprentissage déjà existants pour des réseaux de neurones à impulsions (RNI) afin de proposer une nouvelle procédure adaptée au quantron. Il est immédiat que la découverte d'un algorithme pour le MLQ permettrait d'en exploiter pleinement le potentiel tout en ouvrant la porte à des applications de plus en plus complexes. En explorant cette avenue de recherche, on souhaite également améliorer la compréhension du quantron qui, il faut en être conscient, reste un modèle relativement jeune.

Le reste du mémoire adopte la structure suivante. Le Chapitre 2 passe brièvement en revue les neurones à impulsions, cible un cas particulier du *Spike Response Model* qui ressemble particulièrement au quantron, et identifie l'algorithme SpikeProp qui sert de base au développement d'une nouvelle méthode d'entraînement du MLQ. Avant d'entrer dans la description de cette dernière, le Chapitre 3 discute du choix de la forme de PPS utilisée pour les travaux et établit une propriété importante intégrée au nouvel algorithme. Tandis que le Chapitre 4 décrit en détail la nouvelle méthode, le Chapitre 5 expose et discute des résultats des expérimentations menées sur la reconnaissance de différentes formes. Enfin, le Chapitre 7 revient sur les travaux réalisés, met en évidence les lacunes de la solution proposée et s'en inspire pour proposer de futures avenues de recherche.

CHAPITRE 2 QUANTRON ET NEURONES À IMPULSIONS

L'avènement récent des neurones à impulsions comme nouvelle génération de neurones artificiels prend source dans de nombreuses observations expérimentales en neurosciences rapportant que l'information serait encodée par la valeur des instants des impulsions. C'est le cas notamment dans le système d'écholocation de la chauve-souris où certains neurones sont sensibles au délai entre un signal et sa réverbération (Covey et Casseday, 1999), ainsi que chez le rat pour lequel le temps de la première impulsion après le mouvement des vibrisses joue un rôle critique dans la localisation (Panzeri *et al.*, 2001). La rapidité avec laquelle certains phénomènes se produisent, tels que la reconnaissance d'une image par le système visuel humain (Thorpe et Imbert, 1989), compromet aussi l'idée que la transmission de l'information détaillée soit fondée exclusivement sur l'estimation de taux d'activité du neurone (Gautrais et Thorpe, 1998; Thorpe *et al.*, 2001; VanRullen *et al.*, 2005). Les neurones à impulsions rendent ainsi compte de ces découvertes en manipulant explicitement des trains d'impulsions.

En premier lieu, on expose le formalisme d'un modèle de neurone à impulsions particulier, le *Spike Response Model* (SRM). Une forme particulière du SRM, le neurone SpikeProp, est également présentée étant donné qu'elle partage des caractéristiques d'intérêt avec le quantron. En effet, on établit par la suite un parallèle important entre ces derniers qui permet d'envisager l'apprentissage du quantron en s'inspirant d'un algorithme qui s'applique au neurone SpikeProp. Enfin, on survole rapidement les diverses stratégies d'apprentissage employées pour entraîner les neurones à impulsions. Parmi celles-ci, l'algorithme SpikeProp est détaillé puisqu'il sert de fondation à la nouvelle méthode présentée au Chapitre 4.

2.1 Le *Spike Response Model*

Les neurones à impulsions possèdent tous un potentiel de membrane défini par les impulsions incidentes qui, lorsqu'il atteint un seuil, provoque une impulsion en sortie. Les modèles se regroupent en deux grandes familles selon le mécanisme régissant ce potentiel.

Dans la première, la fonction d'activation respecte une équation différentielle. C'est la forme de cette dernière qui donne naissance à un large éventail de modèles. Par exemple, le neurone *Integrate-and-Fire* avec fuite (LIF) modélise le neurone comme un circuit RC excité par des courants présynaptiques ou externes qui varient dans le temps (Gerstner et Kistler, 2002). Le LIF est en quelque sorte l'archétype de cette famille et se base sur les expérimentations avec des nerfs sciatiques de grenouilles (Lapicque, 1907). Le neurone de Hogdkin-Huxley, pour

sa part, modélise les divers courants ioniques et apporte un réalisme accru au prix d'une implémentation plus lourde. Certains modèles comme ceux de FitzHugh-Nagumo (FitzHugh, 1961) et d'Izhikevich (Izhikevich, 2003) présentent des compromis entre ces deux aspects.

Puisque la fonction d'activation du quantron n'est pas gouvernée par une équation différentielle, il s'avère plus intéressant de travailler avec la seconde famille de neurones, celle du SRM, car le potentiel prend la forme d'une fonction explicite des entrées et des paramètres.

2.1.1 Formalisme du SRM

On considère un neurone j dont les neurones incidents constituent l'ensemble Λ_j. La première contribution au potentiel de membrane provient des trains d'impulsions émis par chaque neurone incident $i \in \Lambda_j$. Un train $S_i(t)$ est représenté par une somme de distributions de Dirac centrées aux temps t_i^* auxquels surviennent les impulsions. L'ensemble des temps des impulsions associées au train $S_i(t)$ est noté $\mathcal{F}_i$, de sorte que $t_i^* \in \mathcal{F}_i$. Cet ensemble est généralement fini.

$$S_i(t) = \sum_{t_i^* \in \mathcal{F}_i} \delta(t - t_i^*) \tag{2.1}$$

Chaque impulsion est associée à un PPS dont la forme est donnée par un noyau $\epsilon(t)$. Ce noyau peut être choisi arbitrairement, mais est en général une exponentielle décroissante, une différence de deux exponentielles ou une fonction α

$$\alpha(t) = \frac{t - \Delta^{\text{ax}}}{\tau^2} \exp\left(-\frac{t - \Delta^{\text{ax}}}{\tau}\right) u(t - \Delta^{\text{ax}}), \tag{2.2}$$

où $\tau > 0$ est une constante de temps déterminant la largeur du signal, $\Delta^{\text{ax}} \geq 0$ est le délai de propagation du signal dans l'axone et $u(\cdot)$ est la fonction échelon (Gerstner, 1995; Maass et Bishop, 2001). L'étendue temporelle du train d'impulsions s'obtient en prenant la convolution avec le noyau. En affectant chaque train d'un poids synaptique w_{ij}, la contribution synaptique des neurones incidents au potentiel de membrane du neurone vaut donc

$$v_j^{\text{syn}}(t) = \sum_{i \in \Lambda_j} \int_{-\infty}^{\infty} \epsilon(t - s) S_i(s) ds = \sum_{i \in \Lambda_j} \sum_{t_i^* \in \mathcal{F}_i} w_{ij} \epsilon(t - t_i^*). \tag{2.3}$$

Il est possible de tenir compte de la période réfractaire du neurone via une fonction $\eta(t)$. Pendant la période réfractaire absolue, le neurone ne peut pas émettre, alors que lors de la période réfractaire relative, le neurone peut émettre mais avec plus de difficulté. À la dernière impulsion émise au temps $\hat{t}_j$, la contribution réfractaire $v_j^{\text{ref}}(t) = \eta(t - \hat{t}_j)$ vient ainsi diminuer

le potentiel de membrane [1]. L'équation complète du potentiel de membrane du neurone SRM est enfin la somme des contributions synaptique et réfractaire [2].

$$v_j(t) = v_j^{\text{ref}}(t) + v_j^{\text{syn}}(t) = \eta(t - \hat{t}_j) + \sum_{i \in \Lambda_j} \sum_{t_i^* \in \mathcal{F}_i} w_{ij} \epsilon(t - t_i^*), \tag{2.4}$$

Le neurone émet une impulsion à chaque instant auquel le potentiel atteint le seuil Γ prédéfini avec une dérivée positive.

$$v_j(t) = \Gamma \ \wedge \ \frac{dv_j(t)}{dt} > 0 \quad \Longrightarrow \quad t \in \mathcal{F}_j \tag{2.5}$$

2.1.2 Le neurone SpikeProp

Certains neurones, comme le neurone SpikeProp (Bohte *et al.*, 2002), sont basés sur le SRM auquel ils apportent certaines simplifications et/ou modifications. D'abord, ce modèle pose une simplification en supposant que chaque neurone n'émet qu'une seule impulsion [3]. La modélisation des périodes réfractaires devient alors superflue de sorte que $v_j^{\text{ref}}(t)$ est identiquement nulle. Ensuite, ce modèle apporte une modification en introduisant K synapses par connexion entre deux neurones. Chacune des synapses transmet l'impulsion du neurone précédent avec ses propres poids $w_{ij}^{(k)}$ et délai $d^{(k)}$ synaptiques. Ces changements mènent au nouveau potentiel de membrane

$$v_j(t) = \sum_{i \in \Lambda_j} \sum_{k=1}^{K} w_{ij}^{(k)} \epsilon(t - t_i^* - d_{ij}^{(k)}) \tag{2.6}$$

qui, lorsqu'il franchit le seuil Γ pour la première fois, génère une unique impulsion t_j^*.

2.1.3 Analogie entre le quantron et le neurone SpikeProp

On constate que le potentiel de membrane du neurone SpikeProp prend une forme similaire à la fonction d'activation du quantron. Le théorème suivant formalise le lien étroit entre ces deux expressions.

---

1. Certains auteurs dont Pfister *et al.* postulent que la fonction réfractaire s'applique à chaque impulsion émise $t_j^* \in \mathcal{F}_j$, de sorte que $v_j^{\text{ref}}(t)$ prend la forme d'une sommation sur les éléments de $\mathcal{F}_j$.

2. Le terme de courant externe est omis. Aussi, le noyau pourrait dépendre de la dernière impulsion émise par le neurone j et on aurait alors une fonction $\epsilon(t - t_i^*, t - \hat{t}_j)$ à deux variables. Cette caractéristique est ici omise et il serait alors plus juste de parler du neurone simplifié SRM_0 (Gerstner et Kistler, 2002). Par souci de concision et parce que toute confusion est improbable, cette distinction n'est pas maintenue dans le reste du mémoire.

3. L'encodage de l'information par le temps de la première impulsion (*time-to-first-spike coding*) a été notamment observé dans les systèmes somatosensoriels du rat et de l'humain (VanRullen *et al.*, 2005).

Théorème 1. *Soit un neurone SpikeProp j recevant une impulsion t_i^* en provenance de chacun de ses M neurones présynaptiques. Soit également un quantron à M entrées. Si les deux neurones partagent le même seuil d'activation Γ, alors sous les équivalences*

$$K \equiv N, \quad w_{ij}^{(k)} \equiv w_{ij}, \quad t_i^* + d_{ij}^{(k)} \equiv \theta_{ij} + (k-1)x_i, \quad \epsilon(t) \equiv \varepsilon_0(t; s_{ij}),$$

la sortie du neurone SpikeProp est identique à la sortie du quantron.

Preuve. Si on injecte les équivalences dans le potentiel de membrane (2.6), on obtient

$$v_j(t) = \sum_{i=1}^{M} \sum_{k=1}^{N} w_{ij}\varepsilon_0(t - \theta_{ij} - (k-1)x_i; s_{ij}) = \sum_{i=1}^{M} \sum_{n=0}^{N-1} w_{ij}\varepsilon_0(t - \theta_{ij} - nx_i; s_{ij}),$$

qui est la fonction d'activation du quantron donnée par (1.4). Puisque le neurone SpikeProp a la même fonction d'activation et le même seuil que le quantron, alors le temps de la première impulsion du premier correspond à la sortie (1.5) du second. □

Ce résultat important offre la possibilité d'adapter au quantron un algorithme d'apprentissage existant s'appliquant au SRM puisque leurs formalismes se rejoignent dans un cas particulier. La section suivante présente l'algorithme SpikeProp identifié à cet effet.

2.2 SpikeProp : base d'un nouvel algorithme

L'apprentissage supervisé de neurones impulsionnels consiste généralement à ajuster les paramètres d'un modèle, la presque totalité du temps les poids synaptiques, afin de reproduire des trains d'impulsions spécifiques. Bien que ces neurones soient plus puissants que le perceptron, ils s'avèrent substantiellement plus complexes à entraîner et il devient beaucoup plus difficile de démontrer rigoureusement des énoncés positifs à ce sujet (Maass et Schmitt, 1999). De ce fait, de nombreuses approches ont été explorées.

Carnell et Richardson ont défini une structure d'espace vectoriel sur les trains d'impulsions pour proposer une méthode approximative pour entraîner un neurone LIF placé en sortie d'une machine à état liquide [4] (LSM) (Carnell et Richardson, 2005).

D'autres méthodes se basent sur le phénomène de *spike-timing-dependent plasticity* (STDP) qui tend à renforcer (resp. affaiblir) les poids lorsqu'une impulsion postsynaptique est émise

4. Une telle machine est composée d'une couche d'entrée, d'un réservoir consistant en un vaste réseau récurrent de neurones connectés aléatoirement qui n'est pas entraîné, et enfin d'une couche de neurones de sortie dont les poids des connexions avec le réservoir sont ajustés (Paugam-Moisy et Bohte, 2012).

juste après (resp. juste avant) une impulsion présynaptique suivant une fenêtre de modification qui décroît exponentiellement avec l'écart temporel entre les impulsions. Ce paradigme rend compte de la potentialisation et de la dépression à long terme (LTP et LTD) observées dans l'hippocampe du rat (Bi et Poo, 1998). Dans la mesure où elle utilise seulement les impulsions et non le modèle de neurone, la STDP peut être utilisée autant avec un LIF (Legenstein *et al.*, 2005) qu'avec un SRM (Strain *et al.*, 2010). Le principal algorithme dans cette catégorie est ReSuMe (Ponulak, 2005) qui parvient à entraîner un neurone de sortie d'une LSM avec une règle de type Widrow-Hoff combinant une partie STDP entre les impulsions présynaptiques et les impulsions cibles, et une partie anti-STDP entre les impulsions pré- et postsynaptiques (Ponulak et Kasiński, 2010). ReSuMe a été agencé à la descente du gradient pour entraîner un réseau à une couche cachée (Grüning et Sporea, 2012) et a été étendu pour inclure l'optimisation de délais synaptiques (Taherkhani *et al.*, 2015).

L'apprentissage statistique a été utilisé avec des neurones stochastiques. Dans ce cas, le potentiel de membrane peut être déterministe et influencer par sa valeur la probabilité de génération d'une impulsion (Barber, 2003; Pfister *et al.*, 2003, 2006). Sinon, l'aspect aléatoire peut être introduit directement dans le potentiel par un terme de bruit et une impulsion est créée quand le seuil est atteint (Pillow *et al.*, 2004). Le but est alors de maximiser un critère faisant intervenir la vraisemblance d'émettre les impulsions cibles [5].

Les stratégies évolutionnistes ont également contribué en considérant les réseaux comme des individus ou des particules. Belatreche *et al.* parviennent à apprendre les poids et les délais de réseaux SpikeProp en considérant une population subissant des mutations génétiques (Belatreche *et al.*, 2003). Également, un algorithme d'évolution différentielle en parallèle basé sur l'évolution indépendente de sous-populations a permis de résoudre le problème du XOR avec un réseau SRM 2-2-1 à six poids (Pavlidis *et al.*, 2005). Enfin, l'optimisation par essaims particulaires (PSO) consiste à déployer un essaim de particules et à faire évoluer chacune d'elles selon une vitesse et une direction déterminées par sa position actuelle et par les meilleures positions atteintes à la fois par la particule et par l'essaim entier (Hong *et al.*, 2010; Vázquez et Garro, 2011).

Il existe un neurone particulier issu du SRM, le tempotron, dont les poids ont été entraînés afin qu'il émette au moins une impulsion en présence d'une première classe de trains d'impulsions et qu'il demeure silencieux lorsque stimulé par la deuxième (Gütig et Sompolinsky,

5. Il est intéressant de noter que cette procédure peut mener à des fenêtres de modification des poids de type STDP (Pfister *et al.*, 2003, 2006).

2006). L'algorithme se base sur la valeur des PPS au maximum atteint par la fonction d'activation et n'a jamais été étendu pour un réseau de tempotrons.

La dernière famille de méthodes repose sur SpikeProp, une adaptation de l'algorithme BP aux RNI. Il permet d'entraîner un réseau de neurones SpikeProp à émettre à différents temps cibles afin, par exemple, de classer les entrées présentées et d'apprendre le problème du XOR encodé temporellement (Bohte *et al.*, 2002) à l'aide d'un neurone d'entrée de référence qui émet toujours à $t = 0$ (Sporea et Gruning, 2011). Des extensions de SpikeProp permettent aux neurones cachés d'émettre plus d'une impulsion et réintroduisent alors la contribution des périodes réfractaires au potentiel (Booij et tat Nguyen, 2005; Ghosh-Dastidar et Adeli, 2009). Une adaptation aux réseaux récurrents a été développée (Tiňo et Mills, 2005), avec des résultats mitigés toutefois. Enfin, l'algorithme *Extended* SpikeProp généralise les règles d'apprentissage aux délais synaptiques $d^{(k)}$ et aux constantes de temps τ des noyaux (Schrauwen et Campenhout, 2004).

Pour choisir une approche particulière à adapter au quantron, il faut considérer ses caractéristiques. D'abord, sa fonction d'activation n'est pas régie par une équation différentielle : les méthodes spécifiques aux neurones de type LIF sont donc écartées. Ensuite, le quantron est un neurone déterministe, ce qui exclut l'apprentissage statistique. Bien sûr, on pourrait considérer un quantron stochastique, mais les approches statistiques ont toutes été appliquées à un seul neurone ce qui s'avère désavantageux si l'on souhaite éventuellement travailler avec des réseaux. On désire travailler avec le quantron en tant qu'outil indépendant d'une structure complexe et coûteuse en calculs telle qu'un réservoir de neurones. Dans cette optique, l'algèbre linéaire et la majorité des méthodes STDP sont exclues. On privilégie également un algorithme qui tire profit du formalisme particulier du quantron. Les méthodes qui ne dépendent pas du neurone utilisé — l'algèbre linéaire, la STDP et les stratégies évolutionnistes — deviennent moins intéressantes d'autant plus que celles du dernier type sont généralement lourdes en calculs.

Ultimement, la caractéristique la plus importante est que pour entraîner le quantron, il faut pouvoir optimiser à la fois les poids, les délais et les largeurs des noyaux. Parmi tous les algorithmes revus, le seul qui s'applique à ces trois classes de paramètres est une extension de SpikeProp. En outre, puisque la méthode permet d'apprendre le temps de la première impulsion, elle rejoint encore plus le formalisme du quantron. C'est pourquoi SpikeProp est retenu comme candidat pour servir d'assise au nouvel algorithme. Le fonctionnement et les performances de SpikeProp sont abordés dans la prochaine sous-section.

2.2.1 L'hypothèse sous-jacente à SpikeProp

La transposition de la BP aux RNI nécessite l'évaluation de la dérivée de la sortie d'un neurone par rapport aux poids. À cette fin, SpikeProp pose l'hypothèse que $v_j(t)$ est localement linéaire autour du temps t_j^* où il atteint le seuil (Bohte *et al.*, 2002), de sorte que

$$\frac{\partial t_j^*}{\partial w_{ij}} = \frac{\partial t_j^*}{\partial v_j(t)} \frac{\partial v_j(t)}{\partial w_{ij}}\bigg|_{t=t_j^*} = \left(-\frac{\partial v_j(t)}{\partial t}\bigg|_{t=t_j^*} \right)^{-1} \frac{\partial v_j(t_j^*)}{\partial w_{ij}}. \tag{2.7}$$

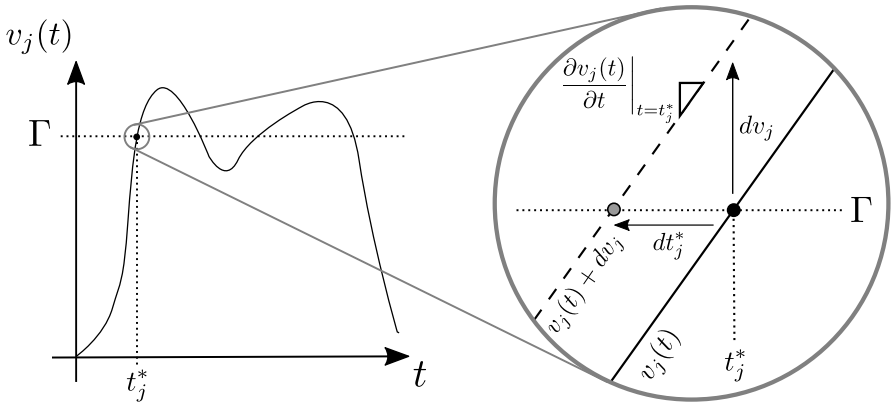
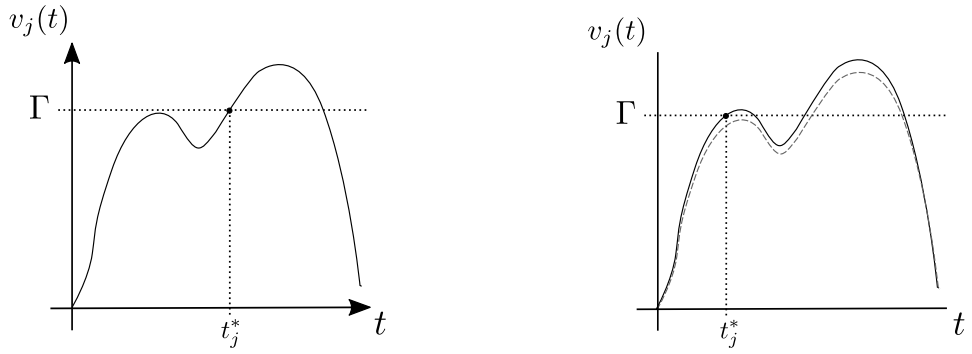


Figure 2.1 Interprétation graphique de l'hypothèse SpikeProp

Concrètement, si $v_j(t)$ est augmentée d'une petite quantité dv_j, le temps t_j^* diminue d'une quantité dt_j^* approximativement égale à $\left(\frac{\partial v_j(t)}{\partial t}\Big|_{t=t_j^*} \right)^{-1} dv_j$ (voir Figure 2.1). Cette hypothèse a été justifiée si $v_j(t)$ est continûment différentiable par rapport au temps et si t_j^* est continue en $v_j(t)$ (Yang *et al.*, 2012). Il arrive toutefois que la seconde condition ne soit pas toujours vérifiée. Par exemple, si un maximum local de $v_j(t)$ survient avant t_j^* et frôle le seuil comme à la Figure 2.2, et que le potentiel est augmenté faiblement mais suffisamment en ce point, alors le maximum local franchira le seuil et le temps de sortie diminuera brusquement (*hair-triggering situation*) tel qu'illustré à la Figure 2.2b. Dans ce cas, la sortie n'est plus une fonction continue du potentiel et des sauts d'erreur surviennent. Ce phénomène est amplifié par la présence simultanée de poids positifs et négatifs favorisant l'apparition de tels maxima (Takase *et al.*, 2009). Ces situations peuvent nuire à la convergence de SpikeProp et certains moyens ont été proposés pour obvier aux divergences parfois observées. Des méthodes d'accélération de la convergence ont été fondées sur un taux d'apprentissage adaptatif (McKennoch *et al.*, 2006; Delshad *et al.*, 2010; Fang *et al.*, 2012; Shrestha et Song, 2015), sur une approximation quadratique locale de la surface d'erreur (McKennoch *et al.*, 2006), sur un terme de *momentum* (Delshad *et al.*, 2010) ou encore sur un terme de régularisation de type



(a) Un minimum local de $v_j(t)$ frôle le seuil

(b) Le minimum local franchit le seuil et réduit brusquement la valeur de t_j^*

Figure 2.2 Situation où le temps de sortie du neurone SpikeProp varie de façon discontinue

weight decay (Yan *et al.*, 2012). Celles-ci s'avèrent toutefois limitées par les irrégularités de la surface d'erreur causées par la discrétisation du temps requise pour implémenter le réseau (Masaru *et al.*, 2008; Thiruvarudchelvan *et al.*, 2013). SpikeProp a aussi été jumelé avec des stratégies évolutives comme la PSO afin d'éviter les minimum locaux de la surface d'erreur (Ahmed *et al.*, 2013).

Une autre lacune de SpikeProp est qu'il suppose que tous les neurones émettent au moins une impulsion. Or, il arrive qu'une ou plusieurs unités restent silencieuses, rendant la méthode inapplicable. Quelques heuristiques ont été proposées pour pallier cette difficulté : initialiser les poids à des valeurs élevées pour s'assurer que la majorité des neurones soient actifs (Bohte *et al.*, 2002; Moore, 2002; McKennoch *et al.*, 2006), augmenter légèrement les poids si le neurone est silencieux (Booij et tat Nguyen, 2005), imposer une valeur minimale aux poids tout au long de l'apprentissage (Wu *et al.*, 2006), diminuer le seuil d'activation (Schrauwen et Campenhout, 2004), ou encore supposer que le neurone émet une impulsion lorsque son potentiel est maximal (Ghosh-Dastidar et Adeli, 2009).

Les lacunes décrites précédemment montrent bien que SpikeProp n'est pas une méthode sans faille. Cela dit, celles-ci ne l'empêchent pas de produire des résultats satisfaisants sur des problèmes à frontières non linéaires et on peut donc envisager de l'utiliser comme base pour un nouvel algorithme. Le problème de divergence dû aux *hair-triggering situations* ne survient pas systématiquement : on obtient en général une solution acceptable moyennant quelques exécutions de SpikeProp. Pour réduire l'impact de cette lacune, la méthode proposée utilise d'ailleurs un taux d'apprentissage adaptatif pour atténuer le risque de divergence.

Par ailleurs, l'incapacité de SpikeProp a gérer les neurones silencieux ne présente pas de problème pour le nouvel algorithme, puisque les heuristiques présentées à la Section 4.5 permettent d'ajuster les paramètres du réseau même lorsqu'un neurone est silencieux. On arrive donc à la conclusion que l'impact des lacunes de SpikeProp sur la méthode proposée est significativement amoindri et que, par conséquent, cette dernière est en mesure d'offrir de bonnes performances sur plusieurs problèmes.

Avant d'entrer dans le détail du fonctionnement du nouvel algorithme (Chapitre 4), le prochain chapitre introduit le noyau triangulaire utilisé pour représenter un PPS dans l'implémentation du quantron. Cette approximation triangulaire plus simple est employée strictement pour des fins de gains de temps de calcul : l'algorithme pourrait ainsi être appliqué avec le noyau original en adaptant les expressions des dérivées du noyau par rapport aux paramètres du réseau. Il ne s'agit donc pas réellement d'une limitation, mais plutôt d'une substitution judicieuse pour gagner en efficacité. En travaillant avec le noyau triangulaire, il est possible de développer une expression analytique de la valeur maximale atteinte par un train de PPS qui sera utilisée dans certaines heuristiques pour neurones silencieux à la Section 4.5.

CHAPITRE 3 PROPRIÉTÉS DU NOYAU TRIANGULAIRE

Dans ce chapitre, on définit le noyau triangulaire employé comme substitut au noyau original $\varepsilon_0(t;s)$ du quantron. Plusieurs raisons motivent ce choix pour la poursuite des travaux. Le noyau original du quantron est lourd à évaluer principalement parce que la fonction $Q(\cdot)$ n'a pas de forme explicite. Pour diminuer le temps de calcul, on utilise donc un noyau plus simple, plus rapide à évaluer et dont la forme rappelle globalement celle de (1.2). Parmi les noyaux carré, parabolique et triangulaire étudiés par (Hackenbeck-Lambert, 2011), le dernier ressort gagnant pour deux raisons. D'abord, la forme de sa dépolarisation est l'inverse de sa repolarisation, comme pour $\varepsilon_0(t;s)$. Ensuite, les images qu'il permet de produire sont semblables à celles générées par le noyau original. En effet, Hackenbeck-Lambert (2011) a comparé des images produites par des réseaux de quantrons utilisant les deux types de noyaux (original et triangulaire) et a observé que les images étaient similaires, ce qui permet de conclure que le noyau triangulaire reproduit fidèlement le comportement du quantron. Le choix du noyau triangulaire se base donc autant sur des motifs d'implémentation informatique que de fidélité de l'approximation.

Puisque le noyau triangulaire possède certaines discontinuités dans ses dérivées premières, la façon de gérer celles-ci est discutée. Ensuite, il est montré qu'un train de PPS est symétrique par rapport à son centre, puis une expression analytique de la valeur maximale atteinte par un tel train est obtenue en fonction de ses paramètres.

3.1 Définition du noyau triangulaire

Le noyau triangulaire est défini par morceaux

$$\varepsilon(t;s) = \begin{cases} t/s & \text{si } 0 \leq t \leq s \\ 2 - t/s & \text{si } s \leq t \leq 2s \\ 0 & \text{autrement} \end{cases}. \tag{3.1}$$

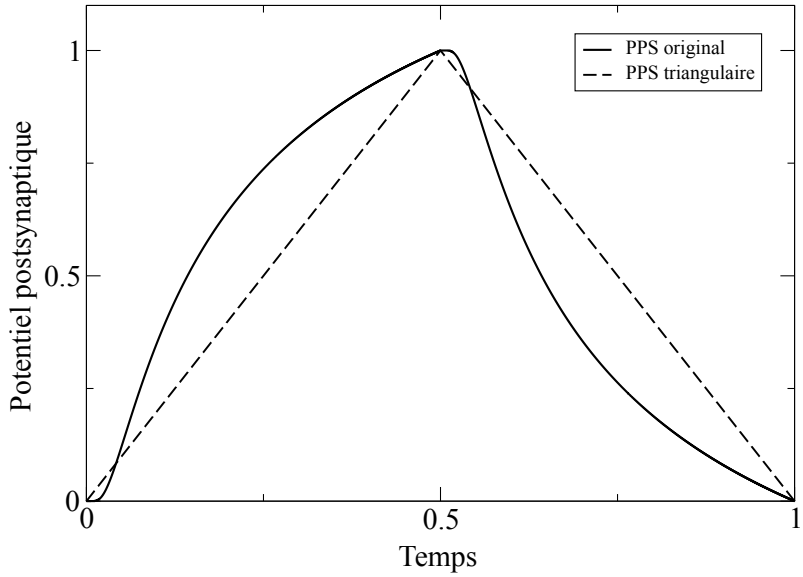
En utilisant la fonction de Heaviside

$$u(t) = \begin{cases} 1 & \text{si } t > 0 \\ \frac{1}{2} & \text{si } t = 0 \\ 0 & \text{autrement} \end{cases}, \tag{3.2}$$

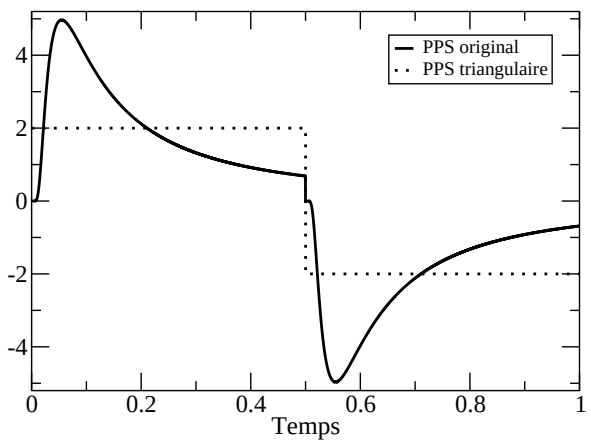
on obtient la définition équivalente

$$\varepsilon(t;s) = \frac{t}{s}\left[u\left(t\right) - u\left(t-s\right)\right] + \left(2 - \frac{t}{s}\right)\left[u\left(t-s\right) - u\left(t-2s\right)\right]. \tag{3.3}$$

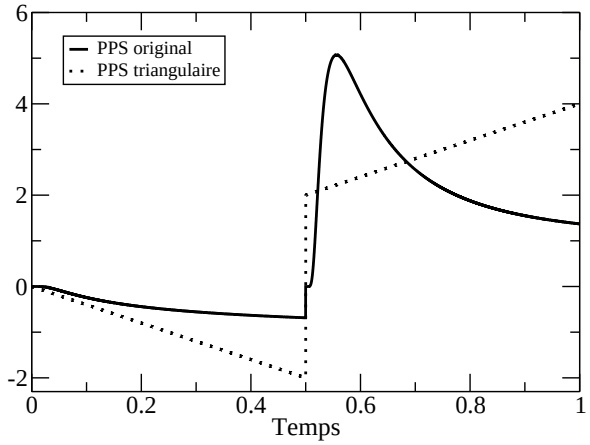
Lorsque la notation $\varepsilon(t;s)$ devient encombrante et qu'aucune ambiguïté n'est possible, elle est réduite à $\varepsilon(t)$ tout simplement. Le noyau triangulaire est comparé au noyau original à la Figure 3.1. Les expressions analytiques des dérivées premières de $\varepsilon_0(t;s)$ par rapport à t et à s utilisées pour tracer les graphiques sont développées à l'Annexe A.



(a) Comparaison des noyaux



(b) Dérivées partielles par rapport à t



(c) Dérivées partielles par rapport à s

Figure 3.1 Noyaux triangulaire $\varepsilon(t;s)$ et original $\varepsilon_0(t;s)$ du quantron ($s = 0{,}5$ et $a = 1{,}5$)

En se concentrant sur l'intervalle $0 < t < s$ (l'autre est symétrique, voir Figure 3.1a), on constate que le noyau triangulaire surestime le noyau original dans une petite région voisine de $t = 0$ tandis qu'il le sous-estime sur le reste de l'intervalle. L'écart absolu maximal entre les noyaux vaut 0,2444 et survient à $t \approx 0{,}2106$, ce qui équivaut à un écart relatif de 37,6 % en prenant la valeur du noyau original comme référence. On peut calculer un écart absolu moyen en évaluant numériquement l'intégrale

$$\frac{1}{s}\int_0^s |\varepsilon(t;s) - \varepsilon_0(t;s)|\, dt \approx 0{,}1469.$$

Puisque les dérivées partielles du noyau seront utilisées par l'algorithme (Figure 3.1b), on compare également les valeurs de celles-ci. La dérivée par rapport au temps est positive et constante pour le noyau triangulaire ; elle est aussi positive pour le noyau original, mais commence à une valeur nulle à $t = 0$, passe par un maximum à $t \approx 0{,}0548$, puis décroît ensuite jusqu'à $t = 0,5$. À ce maximum, qui vaut environ 4,9676, l'écart absolu entre les dérivées des deux noyaux est maximal et vaut 2,9676. Cela correspond à un écart relatif de 59,7 %. En évaluant numériquement l'écart absolu moyen, on obtient 1,1093 qui est bien sûr moins élevé.

On termine en discutant des dérivées partielles par rapport à s. Comme la Figure 3.1c l'illustre, les dérivées sur les intervalles $]0, s[$ et $]s, 2s[$ ne sont pas symétriques : il faut donc considérer l'étendue temporelle du noyau en entier. Sur le premier intervalle, celles-ci sont négatives et décroissantes pour les deux noyaux, mais la dérivée du noyau triangulaire est systématiquement plus grande en magnitude. Sur le second intervalle, les dérivées sont toutes deux positives, mais elles se comportent différemment. Alors que la dérivée du noyau triangulaire est monotone croissante, celle du noyau original augmente rapidement pour atteindre un maximum, puis décroît sur le reste de l'intervalle. L'écart absolu maximal est atteint à $t \approx 0{,}5546$ et vaut 2,8492, soit l'équivalent d'un écart relatif de 56,2 %. Enfin, l'écart absolu moyen calculé sur $[0, 2s]$ est égal à 1,0477.

Cette étude comparative des deux noyaux met en évidence les différences qui les distinguent, différences qui semblent davantage marquées en ce qui a trait aux dérivées partielles. Ce qu'il faut retenir au-delà des écarts maximaux présentés, c'est que les dérivées comparées partagent toujours le même signe. L'algorithme devrait ainsi se comporter similairement en utilisant l'un ou l'autre des noyaux, puisque le signe de l'ajustement de chaque paramètre du réseau de neurones ne dépend pas du noyau utilisé. On rappelle que la solution présentée s'adapterait aisément au noyau original et que le choix du noyau triangulaire est surtout basé sur le gain en efficacité dû à la simplicité de son évaluation.

3.2 Discontinuité des dérivées premières

Bien que le noyau triangulaire soit aisé à évaluer, il faut procéder avec prudence lorsqu'il est question des points où ses dérivées sont discontinues. On explique ici comment les valeurs des dérivées par rapport à t et s sont déterminées. Cette précision s'avère essentielle, car l'algorithme d'apprentissage utilise ces dérivées influenceront son comportement.

3.2.1 Dérivée première par rapport à t

À la Figure 3.1, il est évident que la dérivée première par rapport au temps est discontinue aux temps 0, s et $2s$. Rigoureusement, il est alors impossible de trouver une valeur unique pour la dérivée en ces points. Par contre, on peut procéder en dérivant la définition (3.3) par rapport au temps et en considérant que la dérivée de la fonction de Heaviside est nulle partout (même si en réalité, elle est nulle *presque partout*). On obtient alors

$$\begin{aligned}\varepsilon'(t;s) &= \frac{1}{s}\left[u\left(t\right) - u\left(t-s\right)\right] - \frac{1}{s}\left[u\left(t-s\right) - u\left(t-2s\right)\right] \\ &= \frac{1}{s}\left[u\left(t\right) - 2u\left(t-s\right) + u\left(t-2s\right)\right]. \end{aligned} \tag{3.4}$$

En évaluant l'expression en utilisant (3.2), on trouve

$$\varepsilon'(0;s) = \frac{1}{2s}, \qquad \varepsilon'(s;s) = 0 \quad \text{et} \quad \varepsilon'(2s;s) = -\frac{1}{2s}.$$

La valeur de la dérivée en $t = s$ est ainsi conhérente avec un noyau à *dérivée continue* lorsqu'il atteint son maximum, puisque le test de la dérivée première stipule que la dérivée est nécessairement nulle puisqu'il s'agit d'un extremum de la fonction. Par exemple, on vérifie facilement que la dérivée du noyau $\alpha(t)$ de l'équation (2.2) est nulle au temps $t = \tau + \Delta^{\text{ax}}$ où le noyau atteint son maximum.
La dérivée première de $\varepsilon(t;s)$ par rapport au temps s'écrit donc

$$\varepsilon'(t;s) = \begin{cases} (2s)^{-1} & \text{si } t = 0 \\ s^{-1} & \text{si } 0 < t < s \\ 0 & \text{si } t = s \\ -s^{-1} & \text{si } s < t < 2s \\ -(2s)^{-1} & \text{si } t = 2s \\ 0 & \text{autrement} \end{cases}. \tag{3.5}$$

3.2.2 Dérivée première par rapport à s

On procède de façon analogue en dérivant (3.3) par rapport à s. On obtient alors

$$\begin{aligned}\frac{\partial}{\partial s}\varepsilon(t;s) &= \frac{-t}{s^2}\left[u\left(t\right)-u\left(t-s\right)\right]+\frac{t}{s^2}\left[u\left(t-s\right)-u\left(t-2s\right)\right]\\ &= -\frac{t}{s^2}\left[u\left(t\right)-2u\left(t-s\right)+u\left(t-2s\right)\right].\end{aligned} \tag{3.6}$$

Les valeurs des dérivées aux points de discontinuité sont ainsi définies par

$$\left.\frac{\partial}{\partial s}\varepsilon(t;s)\right|_{t=0}=0, \qquad \left.\frac{\partial}{\partial s}\varepsilon(t;s)\right|_{t=s}=0 \quad \text{et} \quad \left.\frac{\partial}{\partial s}\varepsilon(t;s)\right|_{t=2s}=\frac{1}{s}.$$

Encore une fois, la valeur en $t=s$ est cohérente avec celle du noyau $\alpha(t)$. Pour ce dernier, l'équivalent du paramètre s du quantron est la constante de temps τ qui à la fois détermine la position du maximum du noyau, sépare ses parties croissante et décroissante et définit son étendue temporelle. En dérivant (2.2) par rapport à τ, on trouve

$$\frac{\partial\alpha(t;\tau)}{\partial\tau}=-\frac{t-\Delta^{\mathrm{ax}}}{\tau^3}\left(1-\frac{t-\Delta^{\mathrm{ax}}}{\tau}\right)\exp\left(-\frac{t-\Delta^{\mathrm{ax}}}{\tau}\right)u(t-\Delta^{\mathrm{ax}}) \tag{3.7}$$

qui est négatif pour $t<\tau+\Delta^{\mathrm{ax}}$, positif pour $t>\tau+\Delta^{\mathrm{ax}}$ et qui s'annule en $t=\tau+\Delta^{\mathrm{ax}}$. On constate que ce comportement est similaire à ce qui a été trouvé avec le noyau triangulaire et on a enfin

$$\frac{\partial\varepsilon(t;s)}{\partial s}=\begin{cases}-t/s^2 & \text{si } 0\leq t<s\\ 0 & \text{si } t=s\\ t/s^2 & \text{si } s<t<2s\\ 1/s & \text{si } t=2s\\ 0 & \text{autrement}\end{cases}. \tag{3.8}$$

3.3 Symétrie d'un train de PPS triangulaires

On souhaite ici démontrer qu'un train de PPS triangulaires de fréquence constante est symétrique par rapport à l'axe vertical $t=\vartheta$, où ϑ est défini par la moyenne des temps de début et de fin du train.

$$\vartheta\equiv\frac{1}{2}\left[\theta+\theta+(N-1)x+2s\right]=\theta+\frac{N-1}{2}x+s \tag{3.9}$$

On prend pour acquise la propriété triviale que chaque PPS est symétrique par rapport à son centre situé en $t = s$.

$$\varepsilon(t; s) = \varepsilon(2s - t; s) \tag{3.10}$$

Lemme 1. *Un train de PPS triangulaires de fréquence constante est symétrique par rapport à l'axe vertical passant par son centre.*

Preuve. On considère la valeur du train au temps $\vartheta + t$, où $0 \leq t \leq \vartheta$, c'est-à-dire dans la partie située après son centre.

$$\begin{aligned}
\nu(\vartheta + t) = \sum_{n=0}^{N-1} w\varepsilon(\vartheta + t - \theta - nx) &\overset{(3.10)}{=} \sum_{n=0}^{N-1} w\varepsilon(2s - \vartheta - t + \theta + nx) \\
&= \sum_{n=0}^{N-1} w\varepsilon(\vartheta - t - 2\vartheta + 2s + \theta + nx) \\
&\overset{(3.9)}{=} \sum_{n=0}^{N-1} w\varepsilon\left(\vartheta - t - \theta - [N - 1 - n]x\right)
\end{aligned}$$

En procédant au changement d'indice $n \to N - 1 - n$, on obtient

$$\nu(\vartheta + t) = \sum_{n=0}^{N-1} w\varepsilon(\vartheta - t - \theta - nx) = \nu(\vartheta - t),$$

ce qui complète la démonstration. □

3.4 Valeur maximale atteinte par un train de PPS

Trouver une expression analytique pour la valeur maximale atteinte par une somme de fonctions est un problème généralement difficile. Outre la borne supérieure

$$\max\left[f(t) + g(t)\right] \leq \max f(t) + \max g(t)$$

valable pour deux fonctions f et g majorées, il n'existe pas de formule générale pour la valeur exacte du maximum. Dans le cas présent, la forme spécifique du noyau permet toutefois de trouver une telle expression pour un train de PPS triangulaires en fonction de ses paramètres x, w, s et N. La démonstration de ce résultat requiert quelques lemmes. [1]

1. Dans le raisonnement qui suit, on considère un train de PPS excitateurs associé à un poids positif. Cela dit, les résultats obtenus sont directement applicables à un train de PPS inhibiteurs en remplaçant la notion de maximum par celle de minimum et en donnant une valeur négative à w.

Lemme 2. *La valeur maximale d'un train de N PPS triangulaires de fréquence constante est atteinte en l'un ou plusieurs des N temps* $\theta + nx + s$, $0 \leq n \leq N-1$.

Preuve. Comme établi à la section 3.2, $\varepsilon'(t)$ est discontinue en $t = 0, s, 2s$. Pour $t < 0$ et $t > 2s$, la dérivée est nulle, et elle vaut respectivement s^{-1} et $-s^1$ sur les intervalles $0 < t < s$ et $s < t < 2s$. En conséquence, la dérivée du train de PPS peut être discontinue uniquement aux temps membres de l'ensemble

$$T = \{t_{nh} = \theta + nx + hs \mid 0 \leq n \leq N-1,\ h = 0, 1, 2\}\,.$$

En supposant que les temps de T sont ordonnés du plus petit au plus grand, la dérivée du train entre deux temps consécutifs choisis dans T est constante, du fait que ce sont les seuls points où la dérivée d'un PPS change.

Pour que le maximum du train soit atteint en t, il faut que la dérivée à gauche soit positive et que la dérivée à droite soit négative [2]. À un temps t entre deux temps consécutifs t_1 et $t_2 > t_1$ de T, la dérivée peut être :

i) strictement positive et alors $\nu(t_2) > \nu(t)$;

ii) strictement négative et alors $\nu(t_1) > \nu(t)$;

iii) nulle et alors $\nu(t) = \nu(t_1) = \nu(t_2)$.

Le seul cas où le maximum peut être atteint en $t_1 < t < t_2$ est si la dérivée est nulle et dans ce cas, cette valeur est aussi atteinte en t_1 et en t_2. Ainsi, il s'avère inutile d'évaluer $\nu(t)$ en d'autres points que ceux de T pour identifier le maximum.

En se restreignant à l'ensemble T, alors les seuls temps où la dérivée a la possibilité de diminuer, et donc de passer d'une valeur positive à gauche à une valeur négative à droite, sont les temps t_{n1} où la dérivée d'un noyau passe de s^{-1} à $-s^{-1}$. Autrement, la dérivée d'un noyau augmente en passant de 0 à s^{-1} aux temps t_{n0} et de $-s^{-1}$ à 0 aux temps t_{n2}. Cela démontre que l'étude des seuls temps t_{n1} permet d'identifier le maximum de $\nu(t)$. □

Remarque. Le lemme 2 tire profit du fait que la dérivée entre deux temps consécutifs de T est constante pour des PPS triangulaires. Cela n'est généralement pas le cas pour un noyau quelconque tel que le noyau original du quantron. Par contre, le lemme s'étend à une somme de trains de PPS triangulaires en définissant l'ensemble ordonné T des temps t_{nh} de tous les trains. La dérivée de la somme des trains reste constante entre deux temps consécutifs

2. Ici, les termes « positif » et « négatif » incluent la possibilité d'être nul.

de T de sorte que l'argumentaire reste valide. Il en résulte que la fonction d'activation du quantron atteint sa valeur maximale en au moins un des centres des noyaux $\theta_i + nx_i + s_i$. On exploite ce résultat dans l'implémentation du quantron dans l'environnement MATLAB : la fonction d'activation est uniquement évaluée aux temps t_{nh} et la sortie y est obtenue par interpolation linéaire entre deux évaluations successives. Ceci élimine les erreurs associées à la discrétisation du domaine temporel tout en réduisant avantageusement le temps de calcul.

Les lemmes techniques suivants servent à la démonstration du résultat principal.

Lemme 3. *Soit $N \in \mathbb{N}^*$, $k \in \mathbb{R}_+^*$ et $k^* = \lfloor k \rfloor$ la partie entière*[a] *de k. Alors,*

$$u_n = n + k^* + 1 - \frac{n(n+1) + k^*(k^*+1)}{2k} \leq 2k^* + 1 - \frac{k^*(k^*+1)}{k}$$

pour $0 \leq n \leq k^*$.

a. La partie entière fait ici référence à la partie entière inférieure telle que $k^* \leq k < k^* + 1$.

Preuve. On procède par induction en partant de $n = k^*$. On a

$$u_{k^*} = k^* + k^* + 1 - \frac{k^*(k^*+1) + k^*(k^*+1)}{2k} = 2k^* + 1 - \frac{k^*(k^*+1)}{k}.$$

Supposons que le résultat tienne pour $n = k^* - \ell$, où $0 \leq \ell \leq k^* - 1$. Alors,

$$\begin{aligned} u_{k^*-(\ell+1)} &= 2k^* - \ell - \frac{(k^*-\ell-1)(k^*-\ell) + k^*(k^*+1)}{2k} \\ &= 2k^* - (\ell-1) - \frac{(k^*-\ell)(k^*-\ell+1) + k^*(k^*+1)}{2k} - 1 + \frac{2(k^*-\ell)}{2k} \\ &= u_{k^*-\ell} + \frac{k^*-\ell}{k} - 1 \\ &\leq 2k^* + 1 - \frac{k^*(k^*+1)}{k} \end{aligned}$$

où la dernière ligne utilise l'hypothèse d'induction et le fait que

$$\frac{k^*-\ell}{k} - 1 \leq \frac{k^*}{k} - 1 \leq 1 - 1 = 0. \qquad \square$$

Lemme 4. *Soit N, k et k^* définis comme au Lemme 3 avec la condition supplémentaire*

$$k^* \geq \frac{N-1}{2}.$$

Alors,

$$u_n = N - n + k^* - \frac{(N-1-n)(N-n) + k^*(k^*+1)}{2k} \leq N - \frac{(N-1-k^*)(N-k^*) + k^*(k^*+1)}{2k}$$

pour $k^* \leq n \leq N-1$.

Preuve. On procède encore une fois par induction à partir de $n = k^*$. Pour ce cas, l'inégalité est respectée pusique u_{k^*} est clairement égal à la borne supérieure. Admettons à présent que le résultat soit vrai pour $n = k^* + \ell$, où $0 \leq \ell \leq N-1-k^*$. Alors,

$$\begin{aligned} u_{k^*+(\ell+1)} &= N - \ell - 1 - \frac{(N-1-k^*-\ell-1)(N-k^*-\ell-1) + k^*(k^*+1)}{2k} \\ &= N - \ell - \frac{(N-1-k^*-\ell)(N-k^*-\ell) + k^*(k^*+1)}{2k} - 1 + \frac{2(N-k^*-\ell-1)}{2k} \\ &= u_{k^*+\ell} + \frac{N-1-k^*-\ell}{k} - 1 \\ &\leq N - \frac{(N-1-k^*)(N-k^*) + k^*(k^*+1)}{2k} + \frac{N-1-k^*-\ell}{k} - 1 \end{aligned}$$

où la dernière ligne utilise l'hypothèse d'induction. Sachant que

$$k^* \geq \frac{N-1}{2} \geq \frac{N-1-k^*-\ell}{2},$$

on a

$$\frac{N-1-k^*-\ell}{k} - 1 \leq \frac{k^*}{k} - 1 \leq 1 - 1 = 0$$

ce qui entraîne

$$u_{k^*-(\ell+1)} \leq N - \frac{(N-1-k^*)(N-k^*) + k^*(k^*+1)}{2k}.$$

□

Lemme 5. *Le polynôme quadratique*

$$p(n) = n(n+1) + (N-1-n)(N-n)$$

défini sur les entiers positifs, où $N \in \mathbb{N}^*$, a pour valeur minimale

$$p\left(\frac{N-1}{2}\right) = \frac{N^2-1}{2}$$

si N est impair et

$$p\left(\frac{N}{2}-1\right) = p\left(\frac{N}{2}\right) = \frac{N^2}{2}$$

si N est pair.

Preuve. On développe le polynôme pour obtenir

$$p(n) = 2n^2 - 2(N-1)n + (N-1)N$$

dont le minimum défini sur les réels est situé à

$$n = -\frac{-2(N-1)}{2 \cdot 2} = \frac{N-1}{2}.$$

Si N est impair, alors la solution réelle est également entière et le minimum vaut

$$\begin{aligned} p\left(\frac{N-1}{2}\right) &= 2\frac{(N-1)^2}{4} - 2\frac{(N-1)^2}{2} + (N-1)N \\ &= \frac{(N-1)}{2}\left[2N - (N-1)\right] \\ &= \frac{N^2-1}{2}. \end{aligned}$$

Si N est pair, on pose $N = 2m$ où m est entier de sorte que

$$\frac{N-1}{2} = m - \frac{1}{2},$$

d'où il vient que

$$m - 1 < \frac{N-1}{2} < m.$$

Puisque $m-1$ et m sont à distances égales de part et d'autre de l'abscisse du sommet de la parabole, alors $p(m-1) = p(m)$ et les deux entiers minimisent sa valeur. Par conséquent, la

valeur minimale est

$$p\left(\frac{N}{2}-1\right)=p\left(\frac{N}{2}\right)=2\frac{N^2}{4}-2\frac{(N-1)N}{2}+(N-1)N=\frac{N^2}{2},$$

ce qui achève la démonstration. □

Avec ces résultats préliminaires, il est possible de trouver une expression pour $\max\nu(t)$.

Théorème 2. *Soit $\nu(t)$ un train de N PPS triangulaires espacés de x unités de temps. Alors,*

$$\max\nu(t)=\begin{cases} w\left[2k^*+1-\frac{k^*(k^*+1)}{k}\right] & \text{si } k^*\leq\frac{N-1}{2} \\ w\left[N-\frac{N^2-1}{4k}\right] & \text{si } k^*\geq\frac{N-1}{2} \text{ et } N \text{ est impair}, \\ w\left[N-\frac{N^2}{4k}\right] & \text{si } k^*\geq\frac{N-1}{2} \text{ et } N \text{ est pair} \end{cases} \tag{3.11}$$

où $k=s/x$ et $k^*=\lfloor k\rfloor$ est la partie entière de k. Cette valeur maximale est atteinte pour la première fois en $t=t_{\max}$, où

$$t_{\max}=\begin{cases} \theta+k^*x+s & \text{si } k^*\leq\frac{N-1}{2} \\ \theta+\left(\frac{N-1}{2}\right)x+s & \text{si } k^*\geq\frac{N-1}{2} \text{ et } N \text{ est impair}, \\ \theta+\left(\frac{N}{2}-1\right)x+s & \text{si } k^*\geq\frac{N-1}{2} \text{ et } N \text{ est pair} \end{cases} \tag{3.12}$$

Preuve. Conformément aux Lemmes 1 et 2, la stratégie suivie est d'évaluer l'ensemble des valeurs de $\nu(t_{n1})$ pour $n\leq\frac{N-1}{2}$ afin d'identifier l'indice n qui correspond au maximum du train. On a d'abord

$$\nu(t_{n1})=\sum_{i=0}^{N-1}w\varepsilon(\theta+nx+s-\theta-ix)=w\sum_{i=0}^{N-1}\varepsilon([n-i]x+s). \tag{3.13}$$

Le i-ème noyau contribue

$$\frac{(n-i)x+s}{s}=1+\frac{n-i}{k} \tag{3.14}$$

à la dernière somme si et seulement si

$$0\leq(n-i)x+s\leq s \quad\Longleftrightarrow\quad n\leq i\leq n+k, \tag{3.15}$$

où $k=s/x$. De la même manière, il fournit une contribution de

$$2-\frac{(n-i)x+s}{s}=1+\frac{i-n}{k} \tag{3.16}$$

si et seulement si

$$s \leq (n-i)x + s \leq 2s \quad \Longleftrightarrow \quad n-k \leq i \leq n, \tag{3.17}$$

Puisque i et n sont des entiers, on peut remplacer k par sa partie entière $k^* = \lfloor k \rfloor$ dans les inégalités (3.15) et (3.17). En tenant compte du fait que $0 \leq i \leq N-1$ et en combinant (3.13), (3.14) et (3.16), on obtient

$$\nu(t_{n1}) = w \sum_{i=i_1}^{i_2} 1 - \frac{|n-i|}{k}, \tag{3.18}$$

où

$$i_1 = \max(0, n-k^*) \tag{3.19a}$$

$$i_2 = \min(n+k^*, N-1) \tag{3.19b}$$

Selon les valeurs de N et de k^* du train considéré, il existe quatre cas possibles pour le couple d'indices (i_1, i_2). Toutefois, le Lemme 1 assure que les cas

$$(i_1, i_2) = (0, n+k^*) \quad \text{et} \quad (i_1, i_2) = (n-k^*, N-1)$$

sont équivalents puisque sous le changement d'indice $n \to N-1-n$, i_1 et i_2 s'intervertissent (i_2 devient inférieur ou égal à i_1) et on obtient

$$(0, n+k^*) \to (N-1-[N-1-n+k^*], N-1) = (n-k^*, N-1).$$

Par conséquent, seules les valeurs de $\nu(t_{n1})$ associées aux cas $(0, N-1)$, $(0, n+k^*)$ et $(n-k^*, n+k^*)$ doivent être évaluées. Pour ce faire, on utilise le résultat notoire de la somme des N premiers entiers.

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

On procède en évaluant $\nu(t_{n1})$ pour chacun des cas.

Cas 1 : Si $(i_1, i_2) = (0, N-1)$, alors

$$n-k^* \leq 0 \quad \text{et} \quad n+k^* \geq N-1 \quad \Longleftrightarrow \quad N-1-k^* \leq n \leq k^*. \tag{3.20a}$$

Pour qu'il y ait au moins une valeur de n possible, il faut que

$$N-1-k^* \leq k^* \quad \Longleftrightarrow \quad k^* \geq \frac{N-1}{2}.$$

Dans ce cas,

$$\begin{aligned}\frac{\nu(t_{n1})}{w} &= \sum_{i=0}^{N-1} 1 - \frac{|n-i|}{k} \\ &= N - \frac{1}{k}\sum_{i=-n}^{N-1-n} |i| \\ &= N - \frac{n(n+1)+(N-1-n)(N-n)}{2k}. \end{aligned} \tag{3.20b}$$

Cas 2 : Si $(i_1, i_2) = (0, n+k^*)$, alors

$$n - k^* \leq 0 \quad \text{et} \quad n + k^* \leq N-1.$$

Cela revient à dire que

$$0 \leq n \leq k^* \quad \text{si} \quad k^* \leq \frac{N-1}{2} \tag{3.21a}$$

et que

$$0 \leq n \leq N-1-k^* \quad \text{si} \quad k^* \geq \frac{N-1}{2}. \tag{3.21b}$$

Dans cette situation,

$$\frac{\nu(t_{n1})}{w} = \sum_{i=0}^{n+k^*} 1 - \frac{|n-i|}{k} = n + k^* + 1 - \frac{n(n+1)+k^*(k^*+1)}{2k}. \tag{3.21c}$$

Cas 3 : Si $(i_1, i_2) = (n-k^*, n+k^*)$, alors

$$n - k^* \geq 0 \quad \text{et} \quad n + k^* \leq N-1 \quad \Longleftrightarrow \quad k^* \leq n \leq N-1-k^*. \tag{3.22a}$$

Pour qu'au moins une valeur de n satisfasse ces conditions, il faut que

$$k^* \leq N-1-k^* \quad \Longleftrightarrow \quad k^* \leq \frac{N-1}{2}.$$

Enfin,

$$\frac{\nu(t_{n1})}{w} = \sum_{i=n-k^*}^{n+k^*} 1 - \frac{|n-i|}{k} = 2k^* + 1 - \frac{k^*(k^*+1)}{k}. \tag{3.22b}$$

On constate que les cas 1 et 3 ne peuvent survenir simultanément, ce qui suggère la définition de deux types de trains selon les valeurs de k^* et de N associées.

Trains de type I. Dans le cas où $k^* \leq (N-1)/2$,

$$\frac{\nu(t_{n1})}{w} = \begin{cases} n + k^* + 1 - \frac{n(n+1)+k^*(k^*+1)}{2k} & \text{si } 0 \leq n \leq k^* \\ 2k^* + 1 - \frac{k^*(k^*+1)}{k} & \text{si } k^* \leq n \leq N-1-k^* \\ N - n + k^* - \frac{(N-1-n)(N-n)+k^*(k^*+1)}{2k} & \text{si } N-1-k^* \leq n \leq N-1 \end{cases} . \tag{3.23}$$

L'expression pour $N-1-k^* \leq n \leq N-1$ est obtenue en substituant n par $N-1-n$ dans la première expression conformément au Lemme 1. Le Lemme 3 assure que la première expression est toujours inférieure ou égale à la seconde. Il en découle que

$$\max \nu(t) = w \left[2k^* + 1 - \frac{k^*(k^*+1)}{k} \right].$$

Le premier temps auquel ce maximum est atteint pour $n = k^*$ et donc $t_{\max} = \theta + k^* x + s$.

Trains de type II. Si $k^* \geq (N-1)/2$, alors

$$\frac{\nu(t_{n1})}{w} = \begin{cases} n + k^* + 1 - \frac{n(n+1)+k^*(k^*+1)}{2k} & \text{si } 0 \leq n \leq N-1-k^* \\ N - \frac{n(n+1)+(N-1-n)(N-n)}{2k} & \text{si } N-1-k^* \leq n \leq k^* \\ N - n + k^* - \frac{(N-1-n)(N-n)+k^*(k^*+1)}{2k} & \text{si } k^* \leq n \leq N-1 \end{cases} . \tag{3.24}$$

Le Lemme 1 garantit qu'il est suffisant d'analyser les deuxième et troisième expressions de (3.24). D'une part, le Lemme 4 atteste que la seconde expression évaluée en $n = k^*$ est toujours supérieure ou égale à la troisième. D'autre part, le Lemme 5 permet de choisir les valeurs de n maximisant la deuxième en fonction de la parité de N. Il reste à montrer que ces valeurs optimales sont en accord avec les conditions posées par (3.24).

Si N est impair, la valeur de n optimale est $(N-1)/2$ et elle respecte les conditions puisque

$$N - 1 - k^* \leq N - 1 - \frac{N-1}{2} = \frac{N-1}{2} \leq k^*.$$

L'unique temps auquel ce maximum est atteint est ainsi $t_{\max} = \theta + \left(\frac{N-1}{2}\right) x + s$.
Si N est pair, les valeurs de n optimales sont $N/2-1$ et $N/2$ et elles respectent toutes deux

les conditions, car

$$\begin{aligned} k^* \geq \frac{N-1}{2} \quad &\Longrightarrow \quad k^* \geq \frac{N}{2} \quad \text{et} \quad N-1-k^* \leq \frac{N}{2}-1 \\ &\Longrightarrow \quad N-1-k^* \leq \frac{N}{2}-1 < \frac{N}{2} \leq k^* \end{aligned}$$

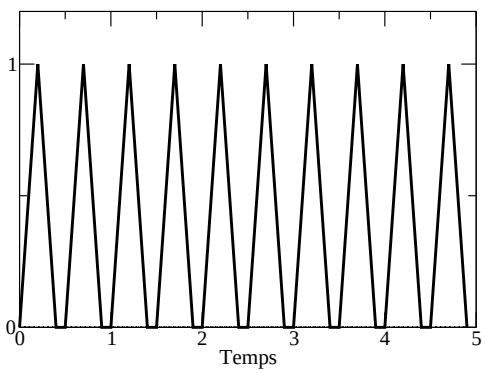
du fait que k^* est entier. Le premier temps où ce maximum est atteint correspond à $n = N/2-1$ et donc $t_{\max} = \theta + \left(\frac{N}{2}-1\right)x + s$.
On obtient finalement

$$\max \nu(t) = \begin{cases} w\left[N - \frac{N^2-1}{4k}\right] & \text{si } N \text{ est impair} \\ w\left[N - \frac{N^2}{4k}\right] & \text{si } N \text{ est pair} \end{cases},$$

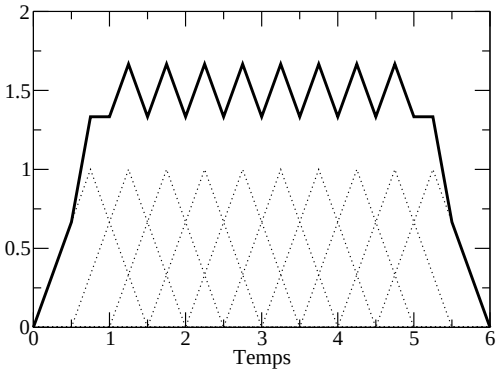
ce qui clôt la démonstration. □

Remarque. Les trains de type I sont ceux qui possèdent suffisamment de PPS pour que la valeur maximale qu'ils atteignent ne soit pas limitée par N, d'où l'absence de ce paramètre dans l'expression finale. Le théorème inclut les trains composés d'une infinité de PPS en les considérant comme étant de type I. Au contraire, l'amplitude maximale des trains de type II est contrainte par le nombre de PPS qui les définit. C'est pourquoi N apparaît dans les résultats qui leur sont associés. La Figure 3.2 illustre les deux situations.

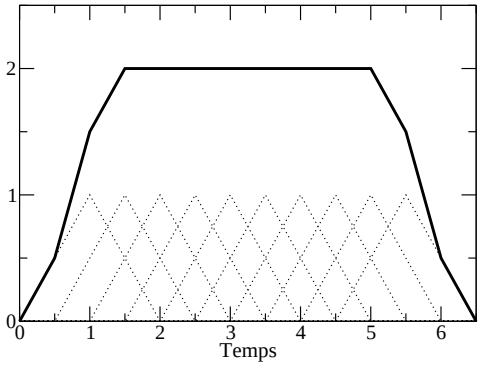
Le résultat du Théorème 2 est intégré à l'algorithme d'apprentissage du quantron exposé au chapitre qui suit. Puisqu'il donne une expression analytique du maximum d'un train de PPS triangulaires, sa dérivée par rapport à s sert à définir des heuristiques pour ajuster les noyaux des quantrons dans un réseau même lorsqu'ils sont silencieux. Plus spécifiquement, le résultat intervient à la Section 4.5.4.



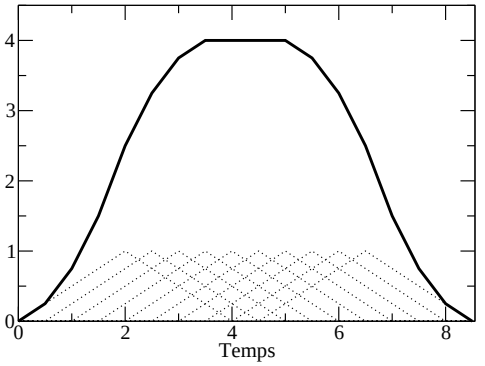
(a) Type I ($s = 0{,}2$; max $= 1$)



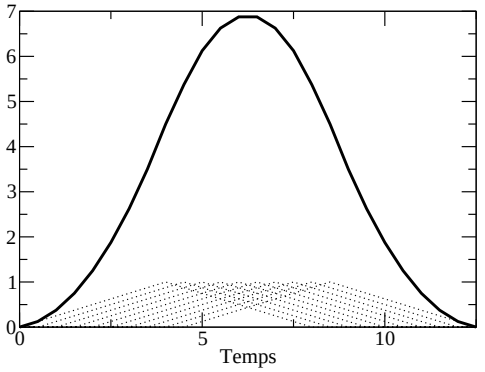
(b) Type I ($s = 0{,}75$; max $= 5/3$)



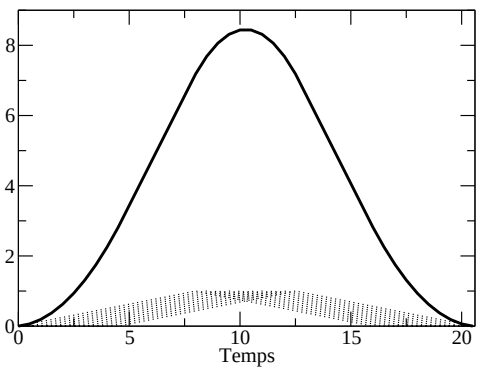
(c) Type I ($s = 1$; max $= 2$)



(d) Type I ($s = 2$; max $= 4$)



(e) Type II ($s = 4$; max $= 55/8$)



(f) Type II ($s = 8$; max $= 135/16$)

Figure 3.2 Valeurs maximales atteintes par des trains de PPS ($x = 0{,}5$, $w = 1$ et $N = 10$)

CHAPITRE 4 ALGORITHME D'APPRENTISSAGE

On définit tout d'abord la tâche de classification que l'on souhaite accomplir à l'aide d'un réseau de quantrons. Par la suite, on établit un nouvel algorithme d'apprentissage basé sur la rétropropagation de l'erreur, l'hypothèse SpikeProp et sur l'expression pour l'extremum d'un train de PPS triangulaires.

4.1 Tâche de classification

On considère des points (x_1, x_2) situés dans le premier quadrant du plan cartésien auxquels sont associés une de deux classes parmi $\mathcal{C}_0$ et $\mathcal{C}_1$. Un point peut ainsi être assimilé au pixel d'une image binaire, pixel qui est éteint s'il appartient à $\mathcal{C}_0$ et allumé autrement. On souhaite alors entraîner un réseau de quantrons à classer correctement les points lorsqu'il est excité par leurs coordonnées. La classe du point est ici encodée par l'état d'activation du quantron de sortie : un pixel éteint correspond à $z = 0$ tandis qu'un pixel allumé équivaut à $z = 1$. L'ensemble d'entraînement pour ce problème est ainsi composé de $\mathcal{N}$ exemples $(x_1^{(n)}, x_2^{(n)})$ associés à autant d'états d'activation cibles $z_d^{(n)}$, $n = 1, \ldots, \mathcal{N}$.

Afin de le rendre plus général, on laisse la possibilité à l'algorithme d'utiliser une valeur cible supplémentaire que l'on affecte à chaque pixel allumé, à savoir le temps $y_d^{(n)}$ auquel le quantron de sortie devrait atteindre le seuil. Le but ultime de la méthode demeure l'apprentissage des états des pixels, mais si les valeurs de y_d sont adéquatement choisies, il s'avère qu'elles rendent possible l'apprentissage de formes sophistiquées autrement très difficile à obtenir. Évidemment, si ces mêmes valeurs sont mal sélectionnées, elles peuvent nuire à l'entraînement du MLQ.

Ainsi, on définit deux types d'erreur quadratique sur l'ensemble d'entraînement prenant la forme d'une somme sur les erreurs individuelles $e^{(n)}$ de chaque exemple. La première est notée E_z et reflète l'objectif fondamental de classification, soit que chaque point appartienne à la bonne classe.

$$E_z = \frac{1}{2} \sum_{n=1}^{\mathcal{N}} \left(e_z^{(n)} \right)^2 = \frac{1}{2} \sum_{n=1}^{\mathcal{N}} \left(z^{(n)} - z_d^{(n)} \right)^2 \tag{4.1}$$

La seconde erreur vise à ce que le temps d'activation du neurone de sortie soit égal au temps cible. Elle est définie seulement pour un neurone de sortie actif lorsqu'un exemple pour lequel $z_d^{(n)} = 1$ lui est présenté. Cette erreur, notée E_y, est équivalente à celle utilisée par l'algorithme

SpikeProp (Bohte *et al.*, 2002).

$$E_y = \frac{1}{2}\sum_{n=1}^{\mathcal{N}} \left(e_y^{(n)}\right)^2 = \frac{1}{2}\sum_{n=1}^{\mathcal{N}} z^{(n)} z_d^{(n)} \left(y^{(n)} - y_d^{(n)}\right)^2 \tag{4.2}$$

À partir de E_y et de E_z, on définit alors l'erreur totale E que l'algorithme d'apprentissage vise à minimiser.

$$E = E_z + E_y = \frac{1}{2}\sum_{n=1}^{\mathcal{N}} \left(z^{(n)} - z_d^{(n)}\right)^2 + \frac{1}{2}\sum_{n=1}^{\mathcal{N}} z^{(n)} z_d^{(n)} \left(y^{(n)} - y_d^{(n)}\right)^2 \tag{4.3}$$

4.2 Modifications au quantron

Pour faciliter le développement de certaines portions de l'algorithme ayant trait aux neurones silencieux, on introduit la variable intermédiaire ξ en s'inspirant librement de l'heuristique utilisée par Ghosh-Dastidar et Adeli (2009) dans leur extension de SpikeProp. On pose

$$\xi = \begin{cases} \arg\max\limits_{t\geq 0} v(t) & \text{si } \max v(t) > 0 \\ \arg\min\limits_{t\geq 0} v(t) & \text{autrement} \end{cases}. \tag{4.4}$$

Ainsi, peu importe l'état d'activité du quantron, si $v(t)$ possède un maximum positif, ξ représente l'instant où ce maximum est atteint. Si $v(t)$ est toujours négative, alors elle traduit plutôt l'instant où $v(t)$ est minimale. Si le maximum ou le minimum global de $v(t)$ (selon le cas) est atteint en plusieurs temps, ξ retourne le temps minimum. Cette nouvelle variable n'ajoute pas de complexité à l'implémentation du MLQ grâce au Lemme 2 qui fournit une méthode rapide pour la calculer.

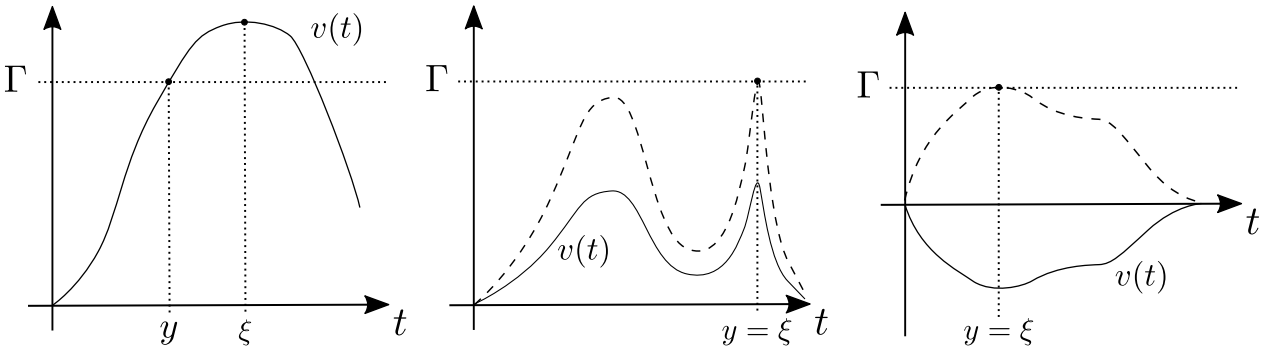
La variable intermédiaire est liée aux sorties y et z du quantron qui ont été définies par (1.5) et (1.6). On rappelle que y est le premier temps auquel la fonction d'activation $v(t)$ atteint le seuil d'activation. Si $v(t)$ n'atteint jamais le seuil, alors on pose $y = \infty$. La sortie z est une variable binaire qui vaut 1 si le seuil est atteint, et 0 dans le cas contraire.

Pour simplifier l'écriture des règles d'apprentissage, on redéfinit ici la sortie y en fonction de ξ lorsque le quantron est silencieux ($z = 0$).

$$y = \begin{cases} \inf\{t > 0 \mid v(t) = \Gamma\} & \text{si } z = 1 \\ \xi & \text{autrement} \end{cases} \tag{4.5}$$

Ainsi, si le quantron est actif, la sortie originale (1.5) est conservée. S'il est inactif, on affecte

à y la valeur de ξ. Diverses situations possibles avec les sorties correspondantes sont illustrées à la Figure 4.1.



(a) $z = 1$ (b) $z = 0$ et $\max v(t) > 0$ (c) $z = 0$ et $\max v(t) \leq 0$

Figure 4.1 Valeur de la variable intermédiaire ξ dans diverses situations

Figure 4.1a : Quantron actif

Lorsque le quantron est actif ($z = 1$), y retourne le premier temps auquel le seuil est atteint. La variable intermédiaire ξ correspond plutôt au premier temps auquel le maximum de la fonction d'activation est atteint.

Figure 4.1b : Quantron silencieux et max $v(t)$ est strictement positif

Le quantron est silencieux ($z = 0$) et il existe un temps pour lequel la fonction d'activation $v(t)$ est strictement positive. La sortie y telle que redéfinie par (4.5) est désormais égale à ξ. Par conséquent, y et ξ représentent toutes deux l'instant où $v(t)$ atteint son *maximum* pour la première fois. C'est à cet instant que la fonction d'activation est le plus près du seuil. Si on augmentait simultanément tous les poids synaptiques des neurones de la couche précédente pour que $v(t)$ atteigne tout juste le seuil (courbe en trait discontinu sur la figure), c'est à cet instant qu'il serait atteint.

Figure 4.1c : Quantron silencieux et max $v(t)$ est négatif (incluant le cas nul)

Le quantron est silencieux ($z = 0$) et la fonction d'activation est toujours négative. Encore une fois, $y = \xi$, car $z = 0$. Dans ce cas spécifique, y et ξ représentent l'instant où $v(t)$ atteint son *minimum* pour la première fois. Si on changeait le signe de tous les poids synaptiques des neurones de la couche précédente pour que $v(t)$ devienne positive et qu'on augmentait ensuite tous les poids simultanément pour que $v(t)$ atteigne tout juste le seuil (courbe en

trait discontinu sur la figure), c'est en $t = \xi$ qu'il serait atteint.

L'ajout de ξ et la modification de y trouvent leur intérêt dans les sections suivantes afin de présenter les règles d'apprentissage et les heuristiques pour neurones silencieux de façon concise. De plus, il est à noter que le changement apporté à y *n'*est *pas* une approximation au quantron exact étant donné que y est modifiée seulement pour les situations où $z = 0$. Or, quand $z = 0$, la sortie y n'a pas d'influence sur les neurones de la couche suivante puisque z multiplie la contribution du neurone silencieux d'après (1.7).

4.3 Règles d'apprentissage

Les règles de la méthode visent à minimiser l'erreur totale par descente du gradient stochastique. Si l'exemple n est présenté à une itération donnée, on tente de minimiser $e_y^{(n)}$ et $e_z^{(n)}$ pour éventuellement parvenir à minimiser E. En dérivant (4.3) par rapport à un paramètre $P \in \{w, \theta, s\}$, on a alors

$$\Delta P^{(n)} \equiv P^{(n+1)} - P^{(n)} = -\eta_P^{(n)} \left(\frac{\partial E}{\partial e_z^{(n)}} \frac{\partial e_z^{(n)}}{\partial P} + \frac{\partial E}{\partial e_y^{(n)}} \frac{\partial e_y^{(n)}}{\partial P} \right), \tag{4.6}$$

où le taux d'apprentissage η_P dépend du type de paramètre mis à jour.

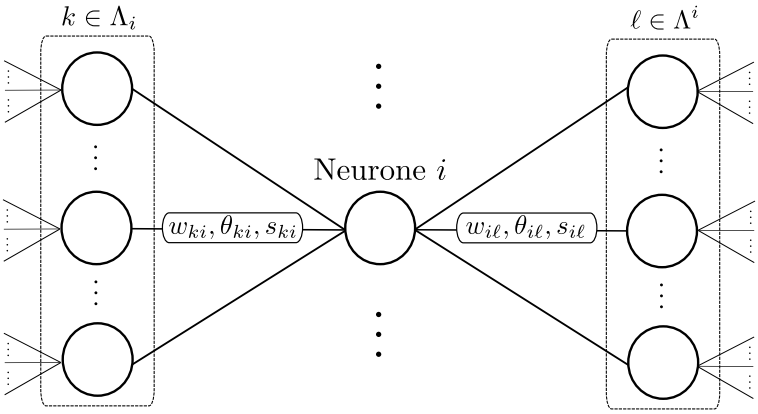


Figure 4.2 Neurone i et ses couches directement en amont (Λ_i) et en aval (Λ^i)

L'expression pour $\Delta P^{(n)}$ est fonction de la position du quantron dans le réseau, selon qu'il est situé dans la couche de sortie ou dans une couche cachée. Dans la suite, on considère ces cas séparément et on omet l'indice n afin d'alléger la notation. Les indices h et j désignent respectivement un neurone caché et un neurone de sortie. De plus, pour un neurone i quelconque, on note Λ_i et Λ^i les ensembles des neurones (ou d'entrées) partageant une synapse avec i qui sont respectivement en amont et en aval de i (voir Figure 4.2).

4.3.1 Gradients locaux

On débute en introduisant quatre quantités analogues au gradient local apparaissant dans l'algorithme BP du MLP (Haykin, 1999). Pour un neurone i, ces gradients locaux sont définis par

$$\delta_{yy,i} \equiv z_k \frac{\partial E}{\partial e_y} \frac{\partial e_y}{\partial y_k} \frac{\partial y_k}{\partial v_k(y_k)} = z_k \frac{\partial E}{\partial e_y} \frac{\partial e_y}{\partial y_k} \left[- \left. \frac{\partial v_k(t)}{\partial t} \right|_{t=y_k} \right]^{-1} \tag{4.7a}$$

$$\delta_{yz,i} \equiv \frac{\partial E}{\partial e_y} \frac{\partial e_y}{\partial z_k} \tag{4.7b}$$

$$\delta_{zy,i} \equiv z_k \frac{\partial E}{\partial e_z} \frac{\partial e_z}{\partial y_k} \frac{\partial y_k}{\partial v_k(y_k)} = z_k \frac{\partial E}{\partial e_z} \frac{\partial e_z}{\partial y_k} \left[- \left. \frac{\partial v_k(t)}{\partial t} \right|_{t=y_k} \right]^{-1} \tag{4.7c}$$

$$\delta_{zz,i} \equiv \frac{\partial E}{\partial e_z} \frac{\partial e_z}{\partial z_k} \tag{4.7d}$$

où l'hypothèse SpikeProp

$$\frac{\partial y_i}{\partial v_i(y_i)} = \left[\frac{\partial v_i(y_i)}{\partial y_i} \right]^{-1} = \left[- \left. \frac{\partial v_i(t)}{\partial t} \right|_{t=y_i} \right]^{-1} \tag{4.8}$$

a été utilisée. La présence des facteurs z_k assure que l'hypothèse SpikeProp est utilisée seulement lorsque le neurone est actif. Ces gradients s'avèrent particulièrement utiles à la Section 4.3.3 où les règles de mise à jour des paramètres des neurones cachés sont développées.

4.3.2 Couche de sortie

La règle de la dérivée en chaîne est utilisée pour exprimer la dérivée de e_y par rapport au paramètre P_{hj} caractérisant la synapse joignant le neurone $h \in \Lambda_j$ et le neurone j.

$$\frac{\partial E}{\partial e_y} \frac{\partial e_y}{\partial P_{hj}} = \frac{\partial E}{\partial e_y} \frac{\partial e_y}{\partial y_j} \frac{\partial y_j}{\partial P_{hj}} = \frac{\partial E}{\partial e_y} \frac{\partial e_y}{\partial y_j} \frac{\partial y_j}{\partial v_j(y_j)} \frac{\partial v_j(y_j)}{\partial P_{hj}} \tag{4.9}$$

On choisit volontairement de ne pas dériver e_y par rapport à z_j dans la dérivée en chaîne, car z_j sert uniquement à déterminer si e_y doit être comptabilisée dans l'erreur totale selon l'état d'activité du neurone j. Si $z_{d,j} = 1$ et que z_j passe de 0 à 1, il est clair que e_z diminue et que e_y augmente. Cela peut engendrer une augmentation globale de l'erreur, mais on ne souhaite pas que les règles d'apprentissage empêchent une telle situation puisqu'elle est souhaitable pour parvenir à minimiser l'erreur de classification.

La première dérivation en chaîne est valide puisque e_y dépend de y_j qui dépend à son tour du paramètre P_{hj}. La seconde dérivation en chaîne provient de l'hypothèse SpikeProp qui

suppose que la sortie dépend de la valeur $v_j(y_j)$ de la fonction d'activation au temps de sortie. On reconnaît alors le gradient local (4.7a) et on obtient

$$\frac{\partial E}{\partial e_y}\frac{\partial e_y}{\partial P_{hj}} = \delta_{yy,j}\frac{\partial v_j(y_j)}{\partial P_{hj}}, \tag{4.10}$$

où

$$\delta_{yy,j} = z_j z_{d,j}\,(y_j - y_{d,j})\left[-\left.\frac{\partial v_j(t)}{\partial t}\right|_{t=y_j}\right]^{-1}. \tag{4.11}$$

Ce gradient local représente la variation de la portion de l'erreur totale associée à e_y de l'exemple en cours lorsque y_j est modifié.
On procède maintenant à l'évaluation de la dérivée de e_z par rapport à P_{hj}.

$$\frac{\partial E}{\partial e_z}\frac{\partial e_z}{\partial P_{hj}} = \frac{\partial E}{\partial e_z}\frac{\partial e_z}{\partial z_j}\frac{\partial z_j}{\partial P_{hj}} \tag{4.12}$$

Les première et seconde dérivées s'obtiennent facilement. Puisque z_j est une variable discrète, on pourrait, comme de Montigny (2014) l'a proposé, directement substituer une approximation continue pour z_j et en déduire une troisième dérivée bien définie. Par contre, cette subtitution altérerait significativement la valeur de $v_j(t)$ et l'on se retrouverait non plus avec le quantron original, mais avec une approximation. On souhaite ici conserver le formalisme original et on choisit plutôt de donner un sens symbolique à cette dérivée. Conséquemment, si une augmentation de P_{hj} tend à rendre le quantron actif (même s'il l'est déjà), alors la dérivée est positive. Au contraire, si ce changement amène le quantron à être silencieux, la dérivée est négative. À la Section 4.5, on propose ainsi des heuristiques de modification des paramètres dans le but de définir cette dérivée le mieux possible.

Pour l'instant, on se contente de noter que

$$\frac{\partial E}{\partial e_z}\frac{\partial e_z}{\partial z_j}\frac{\partial z_j}{\partial P_{hj}} = \delta_{zz,j}\frac{\partial z_j}{\partial P_{hj}}, \tag{4.13}$$

où le gradient local

$$\delta_{zz,j} = z_j - z_{d,j} \tag{4.14}$$

représente l'influence de l'état d'activation du neurone de sortie sur la contribution e_z de l'exemple considéré à l'erreur totale. En retournant à (4.6), on obtient donc la règle suivante pour la couche de sortie.

$$\Delta P_{hj} = -\eta_{P_{hj}}\left(\delta_{yy,j}\frac{\partial v_j(y_j)}{\partial P_{hj}} + \delta_{zz,j}\frac{\partial z_j}{\partial P_{hj}}\right) \tag{4.15}$$

On note bien que les gradients locaux $\delta_{yz,j}$ et $\delta_{zy,j}$ sont nuls pour un neurone de sortie.

4.3.3 Couche cachée

On poursuit en dérivant la règle d'apprentissage pour un paramètre P_{kh} d'une connexion entre un neurone caché h et un des neurones directement en amont $k \in \Lambda_h$. On débute par le terme d'erreur en e_y.

$$\begin{aligned}\frac{\partial E}{\partial e_y}\frac{\partial e_y}{\partial P_{kh}} &= z_h\frac{\partial E}{\partial e_y}\frac{\partial e_y}{\partial y_h}\frac{\partial y_h}{\partial P_{kh}} + \frac{\partial E}{\partial e_y}\frac{\partial e_y}{\partial z_h}\frac{\partial z_h}{\partial P_{kh}} \\ &= \delta_{yy,h}\frac{\partial v_h(y_h)}{\partial P_{kh}} + \delta_{yz,h}\frac{\partial z_h}{\partial P_{kh}}\end{aligned} \tag{4.16}$$

L'introduction du facteur z_h est nécessaire, puisque la dérivée en chaîne par rapport à y_h n'est pas justifiable si le neurone j n'est pas actif. En effet, la valeur de y_h n'influence pas le réseau dans ce cas, comme expliqué à la Section 4.2. Une différence avec la couche de sortie est que l'état d'activation z_h du neurone caché peut influencer indirectement la sortie y_j d'un neurone de sortie via le gradient local $\delta_{yz,h}$, agissant sur e_y du même coup. L'astuce consiste à réécrire les gradients locaux du neurone h en considérant qu'il influence l'erreur via ses connexions avec tous les neurones qui lui succèdent.

$$\begin{aligned}\delta_{yy,h} &= z_h\left(\sum_{\ell\in\Lambda^h} z_\ell\frac{\partial E}{\partial e_y}\frac{\partial e_y}{\partial y_\ell}\frac{\partial y_\ell}{\partial v_\ell(y_\ell)}\frac{\partial v_\ell(y_\ell)}{\partial y_h} + \sum_{\ell\in\Lambda^h}\frac{\partial E}{\partial e_y}\frac{\partial e_y}{\partial z_\ell}\frac{\partial z_\ell}{\partial y_h}\right)\frac{\partial y_h}{\partial v_h(y_h)} \\ &= z_h\left(\sum_{\ell\in\Lambda^h}\delta_{yy,\ell}\frac{\partial v_\ell(y_\ell)}{\partial y_h} + \sum_{\ell\in\Lambda^h}\delta_{yz,\ell}\frac{\partial z_\ell}{\partial y_h}\right)\left[-\left.\frac{\partial v_h(t)}{\partial t}\right|_{t=y_h}\right]^{-1}\end{aligned} \tag{4.17}$$

$$\begin{aligned}\delta_{yz,h} &= \sum_{\ell\in\Lambda^h} z_\ell\frac{\partial E}{\partial e_y}\frac{\partial e_y}{\partial y_\ell}\frac{\partial y_\ell}{\partial v_\ell(y_\ell)}\frac{\partial v_\ell(y_\ell)}{\partial z_h} + \sum_{\ell\in\Lambda^h}\frac{\partial E}{\partial e_y}\frac{\partial e_y}{\partial z_\ell}\frac{\partial z_\ell}{\partial z_h} \\ &= \sum_{\ell\in\Lambda^h}\delta_{yy,\ell}\frac{\partial v_\ell(y_\ell)}{\partial z_h} + \sum_{\ell\in\Lambda^h}\delta_{yz,\ell}\frac{\partial z_\ell}{\partial z_h}\end{aligned} \tag{4.18}$$

L'hypothèse SpikeProp a été employée pour exprimer les dérivées $\partial y_\ell/\partial y_h$ et $\partial y_l/\partial z_h$, la seconde étant encore une fois interprétée symboliquement du fait que z_h est formellement discrète. On constate que $\delta_{yy,h}$ et $\delta_{yz,h}$ s'expriment en fonction de leurs homologues de la couche de neurones suivante calculés préalablement par l'algorithme qui procède en remontant de la couche de sortie vers la couche d'entrée. Les seuls termes restant à définir sont $\partial z_\ell/\partial z_h$ et $\partial z_\ell/\partial y_h$ auxquels on donne aussi une signification symbolique.

Pour définir la première, on suppose que la sortie discrète peut être approchée par la sigmoïde donnée par (1.8). De plus, on approche la dérivée du maximum de $v_\ell(t)$ par rapport à z_h par la dérivée de la fonction d'activation de ℓ par rapport à z_h évaluée en $t = \xi_\ell$ telle que définie par (4.4) à la Section 4.2. On obtient

$$\frac{\partial z_\ell}{\partial z_h} = \lambda\sigma_\ell(1-\sigma_\ell)\left.\frac{\partial v_\ell(t)}{\partial z_h}\right|_{t=\xi_\ell} = \lambda\sigma_\ell(1-\sigma_\ell)\sum_{n=0}^{N-1} w_{h\ell}\varepsilon(\xi_\ell - \theta_{h\ell} - ny_h; s_{h\ell}), \tag{4.19}$$

où

$$\sigma_\ell \equiv \sigma\left(\max v_\ell(t) - \Gamma; \lambda\right). \tag{4.20}$$

Ainsi, si h est actif, la variation que tend à subir z_ℓ si l'on tend à modifier z_h est définie par la valeur du train de PPS de l'entrée h évaluée à la sortie ξ_ℓ. À l'opposé, si h est silencieux, la variation est déterminée par la valeur que prendrait ce même train en ξ_ℓ si h atteignait le seuil en ξ_h. Si cette contribution est positive et que h tend à être plus actif, ℓ tend également à s'activer. Si elle est négative et que h tend à être plus actif, alors ℓ devient moins actif.

De la même façon, on définit la seconde dérivée comme suit

$$\frac{\partial z_\ell}{\partial y_h} = -\lambda\sigma_\ell(1-\sigma_\ell)\sum_{n=0}^{N-1} nz_h w_{h\ell}\varepsilon'(\xi_\ell - \theta_{h\ell} - ny_h; s_{h\ell}), \tag{4.21}$$

de sorte qu'elle peut être non nulle seulement si le neurone h est actif.

À présent, on procède similairement avec la dérivée de l'erreur e_z et on obtient

$$\begin{aligned}\frac{\partial E}{\partial e_z}\frac{\partial e_z}{\partial P_{kh}} &= z_h\frac{\partial E}{\partial e_z}\frac{\partial e_z}{\partial y_h}\frac{\partial y_h}{\partial P_{kh}} + \frac{\partial E}{\partial e_z}\frac{\partial e_z}{\partial z_h}\frac{\partial z_h}{\partial P_{kh}} \\ &= \delta_{zy,h}\frac{\partial v_h(y_h)}{\partial P_{kh}} + \delta_{zz,h}\frac{\partial z_h}{\partial P_{kh}}\end{aligned} \tag{4.22}$$

qui est l'équivalent de (4.16). En développant en dérivée en chaîne sur les neurones de la couche suivante, on obtient les analogues à (4.17) et (4.18).

$$\begin{aligned}\delta_{zy,h} &= z_h\left(\sum_{\ell\in\Lambda^h} z_\ell\frac{\partial E}{\partial e_z}\frac{\partial e_z}{\partial y_\ell}\frac{\partial y_\ell}{\partial v_\ell(y_\ell)}\frac{\partial v_\ell(y_\ell)}{\partial y_h} + \sum_{\ell\in\Lambda^h}\frac{\partial E}{\partial e_z}\frac{\partial e_z}{\partial z_\ell}\frac{\partial z_\ell}{\partial y_h}\right)\frac{\partial y_h}{\partial v_h(y_h)} \\ &= z_h\left(\sum_{\ell\in\Lambda^h}\delta_{zy,\ell}\frac{\partial v_\ell(y_\ell)}{\partial y_h} + \sum_{\ell\in\Lambda^h}\delta_{zz,\ell}\frac{\partial z_\ell}{\partial y_h}\right)\left[-\left.\frac{\partial v_h(t)}{\partial t}\right|_{t=y_h}\right]^{-1}\end{aligned} \tag{4.23}$$

$$\begin{aligned}\delta_{zz,h} &= \sum_{\ell\in\Lambda^h} z_\ell \frac{\partial E}{\partial e_z}\frac{\partial e_z}{\partial y_\ell}\frac{\partial y_\ell}{\partial v_\ell(y_\ell)}\frac{\partial v_\ell(y_\ell)}{\partial z_h} + \sum_{\ell\in\Lambda^h}\frac{\partial E}{\partial e_z}\frac{\partial e_z}{\partial z_\ell}\frac{\partial z_\ell}{\partial z_h} \\ &= \sum_{\ell\in\Lambda^h}\delta_{zy,\ell}\frac{\partial v_\ell(y_\ell)}{\partial z_h} + \sum_{\ell\in\Lambda^h}\delta_{zz,\ell}\frac{\partial z_\ell}{\partial z_h} \end{aligned} \tag{4.24}$$

Enfin, on retourne à (4.6) pour formuler la règle d'apprentissage pour un neurone caché en fonction des gradients locaux (4.17), (4.18), (4.23) et (4.24).

$$\Delta P_{kh} = -\eta_{P_{kh}}\left(\left[\delta_{yy,h}+\delta_{zy,h}\right]\frac{\partial v_h(y_h)}{\partial P_{kh}} + \left[\delta_{yz,h}+\delta_{zz,h}\right]\frac{\partial z_h}{\partial P_{kh}}\right) \tag{4.25}$$

Comme pour l'algorithme de rétropropagation de l'erreur, il s'agit alors d'amorcer l'ajustement des paramètres du réseau de quantrons par la couche de sortie, puis de remonter couche par couche jusqu'à la couche d'entrée.

Si l'on considère un problème d'apprentissage des temps de sortie cibles y_d seulement, l'algorithme peut être appliqué en posant $\delta_{zy} = \delta_{zz} = 0$ pour tous les neurones. Si, de plus, on pose $\delta_{yz} = 0$, alors on retrouve l'algorithme SpikeProp étendu. Au contraire, si on travaille seulement avec les états cibles z_d, alors on fixe $\delta_{yy} = \delta_{yz} = 0$.

La page suivante présente une première synthèse de l'algorithme développé jusqu'ici. Les deux prochaines sections complètent la description de l'algorithme en définissant les dérivées partielles de la fonction d'activation par rapport aux paramètres et aux entrées, ainsi que les dérivées symboliques de l'état d'activation par rapport aux paramètres.

Résumé de l'algorithme (première partie)

Erreur à minimiser

$$E = \frac{1}{2}\sum_{n=1}^{\mathcal{N}}\left(z^{(n)} - z_d^{(n)}\right)^2 + \frac{1}{2}\sum_{n=1}^{\mathcal{N}} z^{(n)} z_d^{(n)}\left(y^{(n)} - y_d^{(n)}\right)^2$$

Modification d'un paramètre d'un neurone de sortie

$$\Delta P_{hj} = -\eta_{P_{hj}}\left(\delta_{yy,j}\frac{\partial v_j(y_j)}{\partial P_{hj}} + \delta_{zz,j}\frac{\partial z_j}{\partial P_{hj}}\right)$$

Gradients locaux d'un neurone de sortie

$$\begin{aligned}
&\delta_{yy,j} = z_j z_{d,j}\left(y_j - y_{d,j}\right)\left[-\left.\tfrac{\partial v_j(t)}{\partial t}\right|_{t=y_j}\right]^{-1} && \delta_{yz,j} = 0\\
&\delta_{zy,j} = 0 && \delta_{zz,j} = z_j - z_{d,j}
\end{aligned}$$

Modification d'un paramètre d'un neurone caché

$$\Delta P_{kh} = -\eta_{P_{kh}}\left(\left[\delta_{yy,h} + \delta_{zy,h}\right]\frac{\partial v_h(y_h)}{\partial P_{kh}} + \left[\delta_{yz,h} + \delta_{zz,h}\right]\frac{\partial z_h}{\partial P_{kh}}\right)$$

Gradients locaux d'un neurone caché

$$\begin{aligned}
\delta_{yy,h} &= z_h\left(\sum_{\ell\in\Lambda^h}\delta_{yy,\ell}\frac{\partial v_\ell(y_\ell)}{\partial y_h} + \sum_{\ell\in\Lambda^h}\delta_{yz,\ell}\frac{\partial z_\ell}{\partial y_h}\right)\left[-\left.\frac{\partial v_h(t)}{\partial t}\right|_{t=y_h}\right]^{-1}\\
\delta_{yz,h} &= \sum_{\ell\in\Lambda^h}\delta_{yy,\ell}\frac{\partial v_\ell(y_\ell)}{\partial z_h} + \sum_{\ell\in\Lambda^h}\delta_{yz,\ell}\frac{\partial z_\ell}{\partial z_h}\\
\delta_{zy,h} &= z_h\left(\sum_{\ell\in\Lambda^h}\delta_{zy,\ell}\frac{\partial v_\ell(y_\ell)}{\partial y_h} + \sum_{\ell\in\Lambda^h}\delta_{zz,\ell}\frac{\partial z_\ell}{\partial y_h}\right)\left[-\left.\frac{\partial v_h(t)}{\partial t}\right|_{t=y_h}\right]^{-1}\\
\delta_{zz,h} &= \sum_{\ell\in\Lambda^h}\delta_{zy,\ell}\frac{\partial v_\ell(y_\ell)}{\partial z_h} + \sum_{\ell\in\Lambda^h}\delta_{zz,\ell}\frac{\partial z_\ell}{\partial z_h}
\end{aligned}$$

Dérivées symboliques de l'état d'activation

$$\begin{aligned}
&\frac{\partial z_\ell}{\partial z_h} = \lambda\sigma_\ell(1-\sigma_\ell)\sum_{n=0}^{N-1} w_{h\ell}\varepsilon(\xi_\ell - \theta_{h\ell} - ny_h; s_{h\ell}) && \sigma_\ell = \sigma\left(\max v_\ell(t) - \Gamma; \lambda\right)\\
&\frac{\partial z_\ell}{\partial y_h} = -\lambda\sigma_\ell(1-\sigma_\ell)\sum_{n=0}^{N-1} n z_h w_{h\ell}\varepsilon'(\xi_\ell - \theta_{h\ell} - ny_h; s_{h\ell})
\end{aligned}$$

4.4 Dérivées partielles de la fonction d'activation

Les dérivées de la fonction d'activation du quantron par rapport aux paramètres et aux entrées sont requises respectivement pour calculer la modification ΔP et les gradients locaux δ_{yy} et δ_{zy}. Elles s'obtiennent en dérivant (1.7) et en utilisant (3.1), (3.5) et (3.8) pour évaluer le noyau et ses dérivées partielles. Pour un neurone i quelconque et $k \in \Lambda_i$, on obtient :

$$\frac{\partial v_i(t)}{\partial w_{ki}} = z_k \sum_{n=0}^{N} \varepsilon(t - \theta_{ki} - nx_k; s_{ki}), \tag{4.26}$$

$$\frac{\partial v_i(t)}{\partial \theta_{ki}} = -z_k w_{ki} \sum_{n=0}^{N} \varepsilon'(t - \theta_{ki} - nx_k; s_{ki}), \tag{4.27}$$

$$\frac{\partial v_i(t)}{\partial s_{ki}} = z_k w_{ki} \sum_{n=0}^{N} \frac{\partial}{\partial s_{ki}} \varepsilon(t - \theta_{ki} - nx_k; s_{ki}), \tag{4.28}$$

$$\frac{\partial v_i(t)}{\partial x_k} = -z_k w_{ki} \sum_{n=0}^{N} n\varepsilon'(t - \theta_{ki} - nx_k; s_{ki}). \tag{4.29}$$

On remarque que pour que la dérivée par rapport à s_{ki} ne soit pas identiquement nulle, le noyau doit dépendre explicitement de s_{ki} et non pas seulement via son domaine de définition s'il est défini par morceaux. Par exemple, la dérivée d'un noyau carré serait identiquement nulle presque partout [1] et ne pourrait être utilisée par cet algorithme.

4.5 Dérivées partielles de l'état d'activation par rapport aux paramètres

On propose ici plusieurs expressions pour les dérivées symboliques

$$\frac{\partial z_i}{\partial w_{ki}}, \quad \frac{\partial z_i}{\partial \theta_{ki}} \quad \text{et} \quad \frac{\partial z_i}{\partial s_{ki}}.$$

Plus précisément, cinq expressions sont développées pour chacune des trois classes de paramètres (w, θ et s), ce qui mène à un total de 15 expressions différentes. Parmi celles-ci, certaines sont générales alors que d'autres s'avèrent plus spécifiques, au sens où elles tentent de tirer profit du rôle particulier du type de paramètre auquel elles s'appliquent. Ces dernières nécessitent de réfléchir à l'effet de chaque paramètre sur l'état d'activation du quantron, c'est-à-dire sur le maximum de sa fonction d'activation. La banque complète d'heuristiques testées est schématisée à la Figure 4.3 afin de bien visualiser sa structure.

1. C'est-à-dire partout sauf sur un ensemble ayant une mesure de Lebesgue nulle. Un tel ensemble sur $\mathbb{R}$ a une « longueur » nulle au sens où il peut être recouvert par l'union d'une famille d'ouverts dont la somme des longueurs peut être choisie arbitrairement petite (Labelle et Mercier, 1993).

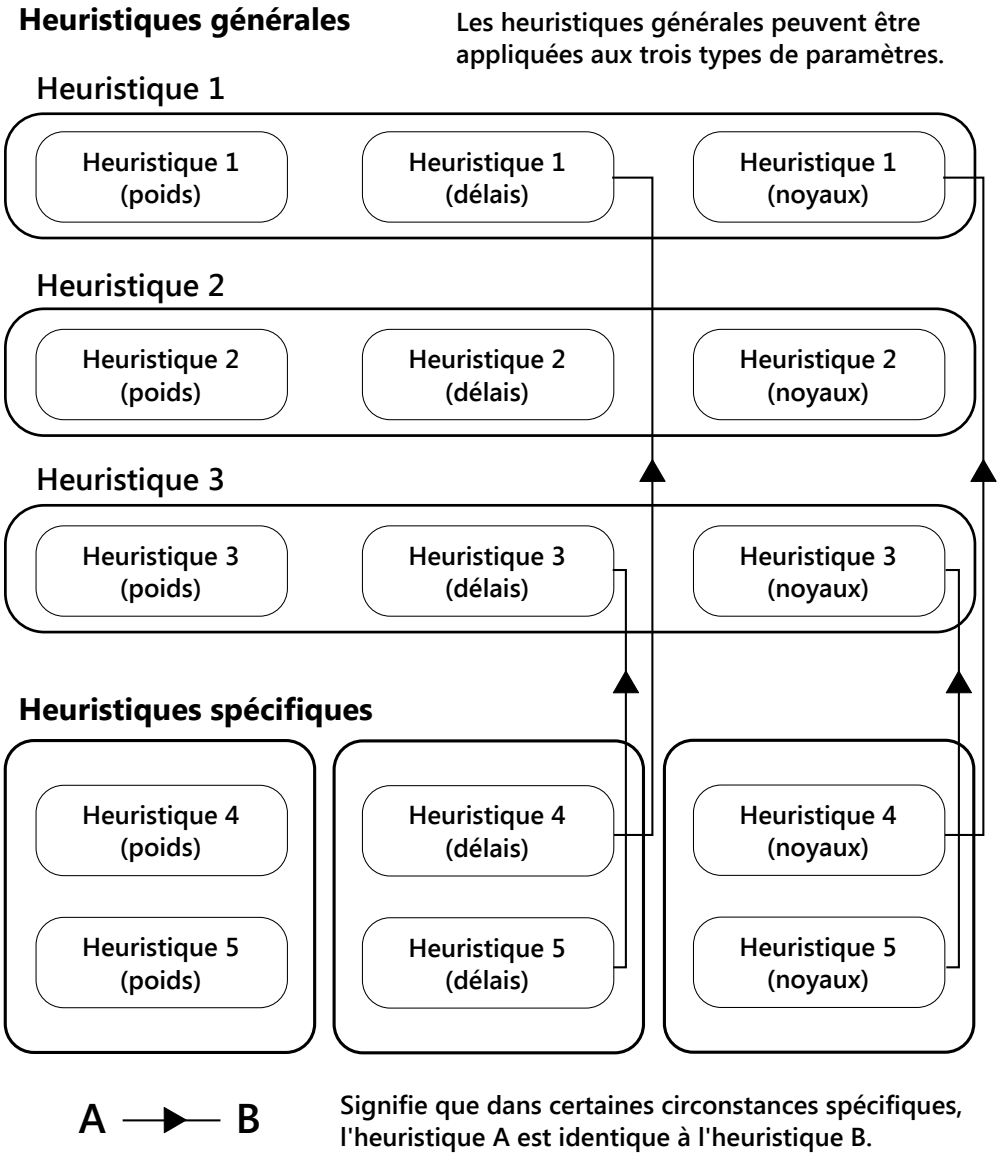


Figure 4.3 Schématisation des 15 heuristiques testées

Évidemment, cette section se veut plutôt exploratoire au sens où les expressions en jeu sont des *heuristiques* qui, pour la plupart du temps, n'émanent pas d'un raisonnement mathématique totalement rigoureux et dont l'efficacité ne peut être démontrée que par l'expérimentation. Au chapitre suivant, chaque expression est donc testée lors de multiples situations d'apprentissage afin d'établir un ensemble de trois heuristiques dit « optimal ».

4.5.1 Heuristiques générales

Les trois premières heuristiques de chaque classe prennent une forme générale.

Heuristique 1 (générale pour w, θ et s)

La dérivée $\partial z_i / \partial P_{ki}$ est approchée par la dérivée de la fonction d'activation en $t = \xi_i$.

$$\frac{\partial z_i}{\partial P_{ki}} = \left.\frac{\partial v_i(t)}{\partial P_{ki}}\right|_{t=\xi_i} \tag{4.30}$$

Cette forme s'inspire de Lastère (2005) qui a utilisé la dérivée au maximum de la fonction d'activation pour modifier les paramètres. L'idée sous-jacente est que si le quantron est silencieux et que $v_i(t)$ possède un maximum strictement positif, le point le plus proche du seuil — et qui est considéré comme étant le plus propice à l'atteindre suite à un changement — est celui où ce maximum est atteint. Si $v_i(t)$ est toujours négative, on évalue la dérivée à son minimum, car le maximum est alors atteint en $t = 0$ où les dérivées sont nulles. On peut interpréter ce choix en affirmant que si les signes de tous les poids étaient modifiés de sorte à les rendre positifs, le minimum de $v_i(t)$ deviendrait le maximum et serait utilisé pour évaluer la dérivée.

Heuristique 2 (générale pour w, θ et s)

La dérivée de la fonction d'activation évaluée en $t = y_i$ sert d'approximation à $\partial z_i / \partial P_{ki}$.

$$\frac{\partial z_i}{\partial P_{ki}} = \left.\frac{\partial v_i(t)}{\partial P_{ki}}\right|_{t=y_i} \tag{4.31}$$

La deuxième heuristique reprend partiellement la première, puisque si $z_i = 0$, $y_i = \xi_i$. Cependant, elle ajoute une touche inspirée de SpikeProp, car si le quantron est actif, on utilise l'information au temps où $v_i(t)$ atteint le seuil. On suppose alors qu'une diminution de l'amplitude de $v_i(t)$ autour du seuil tend à rendre le neurone inactif.

Heuristique 3 (générale pour w, θ et s)

La dérivée de l'état d'activation par rapport au paramètre est donnée par

$$\frac{\partial z_i}{\partial P_{ki}} = \lambda \sigma_i (1 - \sigma_i) \left.\frac{\partial v_i(t)}{\partial P_{ki}}\right|_{t=\xi_i} \tag{4.32}$$

Cette troisième forme provient directement du raisonnement de la Section 4.3.3, plus spécifiquement des équations (4.19) et (4.20). Elle s'obtient si on utilise une sigmoïde pour approcher z_i et si on approche $\partial \max v_i(t)/\partial P_{ki}$ par la dérivée $\partial v_i(t)/\partial P_{ki}$ évaluée en ξ_i. La différence avec l'Heuristique 1 vient du fait que la dérivée est modulée par le facteur $\sigma_i(1-\sigma_i)$ qui est maximal lorsque $\max v_i(t) = \Gamma$ et qui décroît avec l'écart absolu entre le maximum et le seuil.

4.5.2 Poids synaptiques

L'amplitude d'un train de potentiels postsynaptiques est proportionnelle au poids synaptique lui correspondant. La fonction d'activation étant formée par addition de tels trains, il est évident que d'augmenter les poids synaptiques des trains encourage $\max v(t)$ à croître tandis que les diminuer entraîne bien sûr l'effet contraire. Les heuristiques proposées ci-dessous se basent sur ce raisonnement tout en se distinguant sur l'assignation du crédit, c'est-à-dire sur l'importance de la correction apportée au poids d'un train en fonction de sa contribution à la fonction d'activation.

Heuristique 4 (spécifique aux poids w)

La dérivée $\partial z_i/\partial w_{ki}$ est approchée par

$$\frac{\partial z_i}{\partial w_{ki}} = \frac{1}{1 + x_k y_i}. \tag{4.33}$$

Cette heuristique s'inspire de la forme générale de l'apprentissage Hebbien pour lequel la modification apportée au poids dépend des activités pré- et postsynaptiques (Haykin, 1999). La forme la plus simple consiste à modifier le poids proportionnellement au produit des activités. Dans le contexte du quantron, on considère que l'activité est autant plus importante que les PPS sont rapprochés les uns des autres, c'est-à-dire que leur fréquence est élevée. L'activité du neurone est ainsi inversement proportionnelle à sa sortie y_i. Alors, la correction devrait être proportionnelle au produit $(x_k y_i)^{-1}$. Par contre, cette forme n'est pas bornée et peut devenir très grande si l'activité devient très élevée. Pour borner supérieurement ce terme, le produit est augmenté de 1 au dénominateur, ce qui le rend majoré par 1.

Heuristique 5 (spécifique aux poids w)

La dérivée $\partial z_i / \partial w_{ki}$ est posée proportionnelle à l'écart absolu entre le seuil et $v_i(\xi_i)$, ainsi qu'au carré de la contribution $\nu_{ki}(\xi_i)$ du train de l'entrée k.

$$\frac{\partial z_i}{\partial w_{ki}} = \frac{\nu_{ki}^2(\xi_i)}{\sum\limits_{k \in \Lambda_i} \nu_{ki}^2(\xi_i)} \left| \frac{v_i(\xi_i) - \Gamma}{\Gamma} \right| \tag{4.34}$$

La dernière heuristique réunit deux idées intuitives. D'abord, plus le maximum de la fonction d'activation (ou son minimum si elle est toujours négative) est loin du seuil, plus les modifications apportées devraient être importantes, car l'écart à combler est d'autant plus grand pour modifier l'état d'activation. Ensuite, on émet une hypothèse quant à l'assignation du crédit en supposant que plus le train de l'entrée k contribue en valeur absolue à l'atteinte de l'extremum de $v_i(t)$, plus il doit être corrigé fortement.

4.5.3 Délais synaptiques

Le rôle des délais dans l'atteinte du maximum de la fonction d'activation est plus subtil que celui des poids, étant donné qu'ils contrôlent la synchronisation des trains de PPS. Par exemple, pour deux trains excitateurs de paramètres identiques, le déphasage peut faire varier l'amplitude maximale du simple au double. Pour plus de deux trains quelconques, les interactions deviennent hautement complexes et développer une expression analytique pour cette dernière est impraticable.

Comme la Section 2.2 en fait foi, rares sont les algorithmes pour RNI qui proposent des règles d'apprentissage pour les délais synaptiques. La plasticité synaptique semble très souvent réduite à l'efficacité des synapses et donc aux poids exclusivement. Pourtant, il a été observé récemment que les délais peuvent aussi être modifiés et contribuer par le fait même à la plasticité synaptique (Lin et Faber, 2002). Par contre, cette plasticité ne serait pas régie par une loi de type Hebbien (Senn *et al.*, 2002).

À défaut de pouvoir s'inspirer d'une règle existante ou d'un principe biologique, on propose le raisonnement suivant qui émane des heuristiques générales déjà présentées. Analysons le cas possible où un seul train de PPS, d'indice k_1, contribue à $v_i(\xi_i)$. Cela revient à dire que l'ensemble

$$X = \{k \ \text{ tel que } \ |\nu_{ki}(\xi_i)| > 0 \ \text{ et } \ \nu'_{ki}(\xi_i) = 0\} \tag{4.35}$$

est de cardinalité $|X| = 1$ et contient uniquement k_1. On rappelle que la dérivée

$$\left.\frac{\partial v_i(t)}{\partial \theta_{ki}}\right|_{t=\xi_i}$$

est proportionnelle à la dérivée de $\nu_{ki}(t)$ par rapport à t selon (4.27). Dans la situation considérée, elle s'annule pour tout k, incluant k_1. En effet, pour $k \neq k_1$, la dérivée de $\nu_{ki}(t)$ est nulle partout où $\nu_{ki}(t) = 0$, ce qui inclut $t = \xi_i$ par hypothèse. Pour $k = k_1$, on a $v_i(\xi_i) = \nu_{k_1 i}(\xi_i)$ et la dérivée par rapport au temps y est donc nulle puisqu'il s'agit d'un extremum.

Par conséquence, aucun délai n'est modifié dans un tel cas. Pour corriger cette inaction, on propose de modifier uniquement $\theta_{k_1 i}$ pour le rapprocher du train de PPS qui en est le plus près si cela encourage une diminution de l'erreur [2]. On définit donc k_2 le train voisin à k_1 par

$$k_2 = \underset{\substack{k \neq k_1 \\ z_k = 1}}{\arg\min} \left| \vartheta_k - \vartheta_{k_1} \right|, \tag{4.36}$$

où ϑ est le centre d'un train tel que défini par (3.9). Évidemment, l'heuristique est appliquée seulement si k_2 existe. La distance algébrique $\Delta\vartheta$ entre k_1 et k_2 est ainsi donnée par

$$\Delta\vartheta = \vartheta_{k_2} - \vartheta_{k_1}, \tag{4.37}$$

cette distance étant positive si le centre de k_1 est situé avant celui de k_2, et négative dans le cas contraire. On stipule alors que rapprocher θ_{k_1} de θ_{k_2} favorise les interactions entre les trains et tend à atteindre l'état d'activation souhaité pour le quantron. Puisqu'on autorise seulement les trains à se rapprocher, on doit distinguer deux cas.

1. Si $\delta_{yz,h} + \delta_{zz,h} > 0$, alors il faut diminuer l'activité du quantron pour réduire l'erreur. Le train k_1 est alors déplacé vers k_2 seulement si $w_{k_2} < 0$;
2. Si $\delta_{yz,h} + \delta_{zz,h} < 0$, c'est qu'il est au contraire préférable d'accroître l'activité du quantron. Dans ce cas, on rapproche les trains seulement si $w_{k_2} > 0$.

Il reste à définir l'amplitude de la dérivée. On recourt ici au hasard en utilisant une loi

2. La possibilité d'éloigner le train a été considérée, mais elle a été écartée car elle menait souvent à des divergences. Les procédures d'éloignement repoussaient parfois indéfiniment un train, l'isolant des autres et allant à l'encontre de l'objectif de l'heuristique qui vise à favoriser les interactions entre trains.

exponentielle [3] $\text{Exp}(\mu^{-1})$ de paramètre

$$\mu^{-1} = -\frac{\ln(p)}{|\Delta\vartheta|}, \tag{4.38}$$

où $0 < p < 1$ est la probabilité que $\text{Exp}(\mu^{-1}) > |\Delta\vartheta|$, soit la distance entre les centres des trains. La loi exponentielle a été choisie parce qu'elle a pour support les réels positifs, qu'elle se définit simplement à l'aide d'un seul paramètre et que sa fonction de densité est monotone décroissante de sorte que plus un nombre est grand, moins il a de chance d'être généré. On choisit p petite, mais non nulle, afin de permettre que l'ordre temporel des centres des trains puisse être modifié lorsqu'ils sont rapprochés l'un de l'autre. Puisque $\text{E}[\text{Exp}(\mu^{-1})] = \mu$, l'amplitude moyenne est d'autant plus grande que p tend vers 1 et que la distance entre les deux trains voisins est élevée. En pratique, plusieurs valeurs de p ont été testées, mais la valeur $p = 0{,}01$ a produit les meilleurs résultats.

Les deux dernières heuristiques sont ainsi des modifications des Heuristiques 1 et 3. Elles se formulent de façon concise en utilisant le symbole de Kronecker δ^{ki} valant 1 si $k = i$ et 0 autrement, et la fonction signe $\text{sgn}(x)$ qui retourne 1 si $x > 0$, -1 si $x < 0$ et 0 si $x = 0$.

Heuristique 4 (spécifique aux délais θ)

Si $|X| > 1$, l'Heuristique 1 est employée. Dans le cas contraire, la dérivée est proportionnelle à une loi exponentielle de moyenne μ définie par (4.38).

$$\frac{\partial z_i}{\partial \theta_{ki}} = \begin{cases} \left.\dfrac{\partial v_i(t)}{\partial \theta_{ki}}\right|_{t=\xi_i} & \text{si } |X| > 1 \\ \delta^{k_1 k}\,\text{sgn}\,(\Delta\vartheta) u\left(-\left[\delta_{yz,h} + \delta_{zz,h}\right] w_{k_2}\right) \text{Exp}\,(\mu^{-1}) & \text{si } |X| = 1 \end{cases} \tag{4.39}$$

Heuristique 5 (spécifique aux délais θ)

L'Heuristique 3 est appliquée si $|X| > 1$. Autrement, la dérivée est proportionnelle à une loi exponentielle de moyenne μ définie par (4.38).

$$\frac{\partial z_i}{\partial \theta_{ki}} = \begin{cases} \lambda\sigma_i(1-\sigma_i) \left.\dfrac{\partial v_i(t)}{\partial \theta_{ki}}\right|_{t=\xi_i} & \text{si } |X| > 1 \\ \delta^{k_1 k}\,\text{sgn}\,(\Delta\vartheta) u\left(-\left[\delta_{yz,h} + \delta_{zz,h}\right] w_{k_2}\right) \text{Exp}\,(\mu^{-1}) & \text{si } |X| = 1 \end{cases} \tag{4.40}$$

3. La loi exponentielle $\text{Exp}(\lambda)$ de paramètre λ a pour fonction de densité $f(x) = \lambda \exp(-\lambda x)$ où $x \in [0, \infty)$. Sa moyenne est égale à λ^{-1}.

4.5.4 Largeurs de noyaux

À l'instar des délais, le rôle des largeurs des PPS s'avère bien plus complexe que celui des poids. En excluant SpikeProp étendu (Schrauwen et Campenhout, 2004), aucun algorithme d'apprentissage ne présente de règles pour les optimiser. On sait des neurosciences que la morphologie et la densité de canaux ioniques de l'arbre dendritique (Gulledge *et al.*, 2005) ainsi que les propriétés de la synapse (contenu des vésicules présynaptiques, nombre de NT relâchés, densité et cinétique des récepteurs postsynaptiques) (Magee, 2000) sont autant de facteurs jouant aussi sur la largeur d'un PPS, qui peut prendre des valeurs de 1 ms à 50 ms selon le type de neurone étudié (voir Jack *et al.* (1971); Iansek et Redman (1973); Fetz et Gustafsson (1983); Hamm *et al.* (1987); Sayer *et al.* (1990); Williams et Stuart (2000)). Il s'avère par contre difficile de tirer des connaissances actuelles un principe de base pour l'élaboration d'une règle d'apprentissage.

On examine donc encore une fois le cas où $|X| = 1$. Un raisonnement analogue à celui suivi pour les délais montre que les Heuristiques 1 et 3 mènent au statu quo. Pour éviter cette stagnation, on utilise le Théorème 2 qui donne une expression analytique de l'amplitude maximale d'un train de PPS. En dérivant (3.11) par rapport à s, on obtient

$$\frac{\partial}{\partial s}\max \nu(t) = \begin{cases} w\frac{k^*(k^*+1)}{k^2 x} & \text{si } k^* \leq \frac{N-1}{2} \\ w\frac{N^2-1}{4k^2 x} & \text{si } k^* \geq \frac{N-1}{2} \text{ et } N \text{ est impair} \\ w\frac{N^2}{4k^2 x} & \text{si } k^* \geq \frac{N-1}{2} \text{ et } N \text{ est pair} \end{cases} \tag{4.41}$$

en utilisant le fait que la dérivée partielle de k^* par rapport à s est nulle presque partout.

Heuristique 4 (spécifique aux largeurs de noyaux s)

On utilise l'Heuristique 1 si $|X| > 1$. Autrement, l'expression (4.41) définit la dérivée, non nulle pour l'unique train k_1 contribuant à l'extremum de la fonction d'activation.

$$\frac{\partial z_i}{\partial s_{ki}} = \begin{cases} \left.\frac{\partial v_i(t)}{\partial s_{ki}}\right|_{t=\xi_i} & \text{si } |X| > 1 \\ \delta_{k_1 k}\frac{\partial}{\partial s_{ki}}\max \nu_{ki}(t) & \text{si } |X| = 1 \end{cases} \tag{4.42}$$

Heuristique 5 (spécifique aux largeurs de noyaux s)

On emploie l'Heuristique 3 si $|X| > 1$. Sinon, l'expression (4.41) définit la dérivée.

$$\frac{\partial z_i}{\partial s_{ki}} = \begin{cases} \lambda\sigma_i(1-\sigma_i)\left.\dfrac{\partial v_i(t)}{\partial s_{ki}}\right|_{t=\xi_i} & \text{si } |X| > 1 \\ \delta_{k_1 k}\dfrac{\partial}{\partial s_{ki}}\max \nu_{ki}(t) & \text{si } |X| = 1 \end{cases} \tag{4.43}$$

La seconde partie du résumé de l'algorithme fait l'objet de l'encadré à la page suivante. Au chapitre suivant, on évalue la performance de chacune des heuristiques afin de retenir la meilleure pour chaque classe de paramètres.

Résumé de l'algorithme (seconde partie)

Dérivées partielles de la fonction d'activation

$$\frac{\partial v_i(t)}{\partial w_{ki}} = z_k \sum_{n=0}^{N} \varepsilon(t-\theta_{ki}-nx_k;s_{ki}) \qquad \frac{\partial v_i(t)}{\partial \theta_{ki}} = -z_k w_{ki} \sum_{n=0}^{N} \varepsilon'(t-\theta_{ki}-nx_k;s_{ki})$$
$$\frac{\partial v_i(t)}{\partial s_{ki}} = z_k w_{ki} \sum_{n=0}^{N} \frac{\partial}{\partial s_{ki}}\varepsilon(t-\theta_{ki}-nx_k;s_{ki}) \qquad \frac{\partial v_i(t)}{\partial x_k} = -z_k w_{ki} \sum_{n=0}^{N} n\varepsilon'(t-\theta_{ki}-nx_k;s_{ki})$$

Heuristiques générales ($P \in \{w, \theta, s\}$)

$$\text{(H1)} \quad \frac{\partial z_i}{\partial P_{ki}} = \left.\frac{\partial v_i(t)}{\partial P_{ki}}\right|_{t=\xi_i} \qquad \text{(H2)} \quad \frac{\partial z_i}{\partial P_{ki}} = \left.\frac{\partial v_i(t)}{\partial P_{ki}}\right|_{t=y_i}$$
$$\text{(H3)} \quad \frac{\partial z_i}{\partial P_{ki}} = \lambda\sigma_i(1-\sigma_i) \left.\frac{\partial v_i(t)}{\partial P_{ki}}\right|_{t=\xi_i}$$

Heuristiques spécifiques à w

$$\text{(H4)} \quad \frac{\partial z_i}{\partial w_{ki}} = \frac{1}{1+x_k y_i} \qquad \text{(H5)} \quad \frac{\partial z_i}{\partial w_{ki}} = \frac{\nu_{ki}^2(\xi_i)}{\sum\limits_{k\in\Lambda_i} \nu_{ki}^2(\xi_i)} \left|\frac{v_i(\xi_i)-\Gamma}{\Gamma}\right|$$

Heuristiques spécifiques à θ

$$\text{(H4)} \quad \frac{\partial z_i}{\partial \theta_{ki}} = \begin{cases} \left.\dfrac{\partial v_i(t)}{\partial \theta_{ki}}\right|_{t=\xi_i} & \text{si } |X|>1 \\ \delta^{k_1 k}\,\mathrm{sgn}\,(\Delta\vartheta) u\left(-\left[\delta_{yz,h}+\delta_{zz,h}\right] w_{k_2}\right)\mathrm{Exp}\,(\mu) & \text{si } |X|=1 \end{cases}$$
$$\text{(H5)} \quad \frac{\partial z_i}{\partial \theta_{ki}} = \begin{cases} \lambda\sigma_i(1-\sigma_i) \left.\dfrac{\partial v_i(t)}{\partial \theta_{ki}}\right|_{t=\xi_i} & \text{si } |X|>1 \\ \delta^{k_1 k}\,\mathrm{sgn}\,(\Delta\vartheta) u\left(-\left[\delta_{yz,h}+\delta_{zz,h}\right] w_{k_2}\right)\mathrm{Exp}\,(\mu) & \text{si } |X|=1 \end{cases}$$

Heuristiques spécifiques à s

$$\text{(H4)} \quad \frac{\partial z_i}{\partial s_{ki}} = \begin{cases} \left.\dfrac{\partial v_i(t)}{\partial s_{ki}}\right|_{t=\xi_i} & \text{si } |X|>1 \\ \delta_{k_1 k} \dfrac{\partial}{\partial s_{ki}} \max \nu_{ki}(t) & \text{si } |X|=1 \end{cases}$$
$$\text{(H5)} \quad \frac{\partial z_i}{\partial s_{ki}} = \begin{cases} \lambda\sigma_i(1-\sigma_i) \left.\dfrac{\partial v_i(t)}{\partial s_{ki}}\right|_{t=\xi_i} & \text{si } |X|>1 \\ \delta_{k_1 k} \dfrac{\partial}{\partial s_{ki}} \max \nu_{ki}(t) & \text{si } |X|=1 \end{cases}$$

CHAPITRE 5 PRÉSENTATION DES RÉSULTATS

Dans ce chapitre, on applique l'algorithme sur des formes variées représentant des problèmes de classification binaire. D'abord, on utilise une sélection d'images générées par un quantron pour mettre à l'épreuve les heuristiques de la Section 4.5 et en extraire un sous-ensemble optimal. Puisque les images ont été produites par un quantron, on utilise les deux sorties cibles y_d et z_d pour l'apprentissage. Ensuite, avec les heuristiques optimales, on recourt à la méthode pour résoudre d'autres problèmes tels que le XOR. Ceux-ci n'ayant pas été générés par un MLQ, on utilise seulement les états d'activation cibles z_d pour l'entraînement.

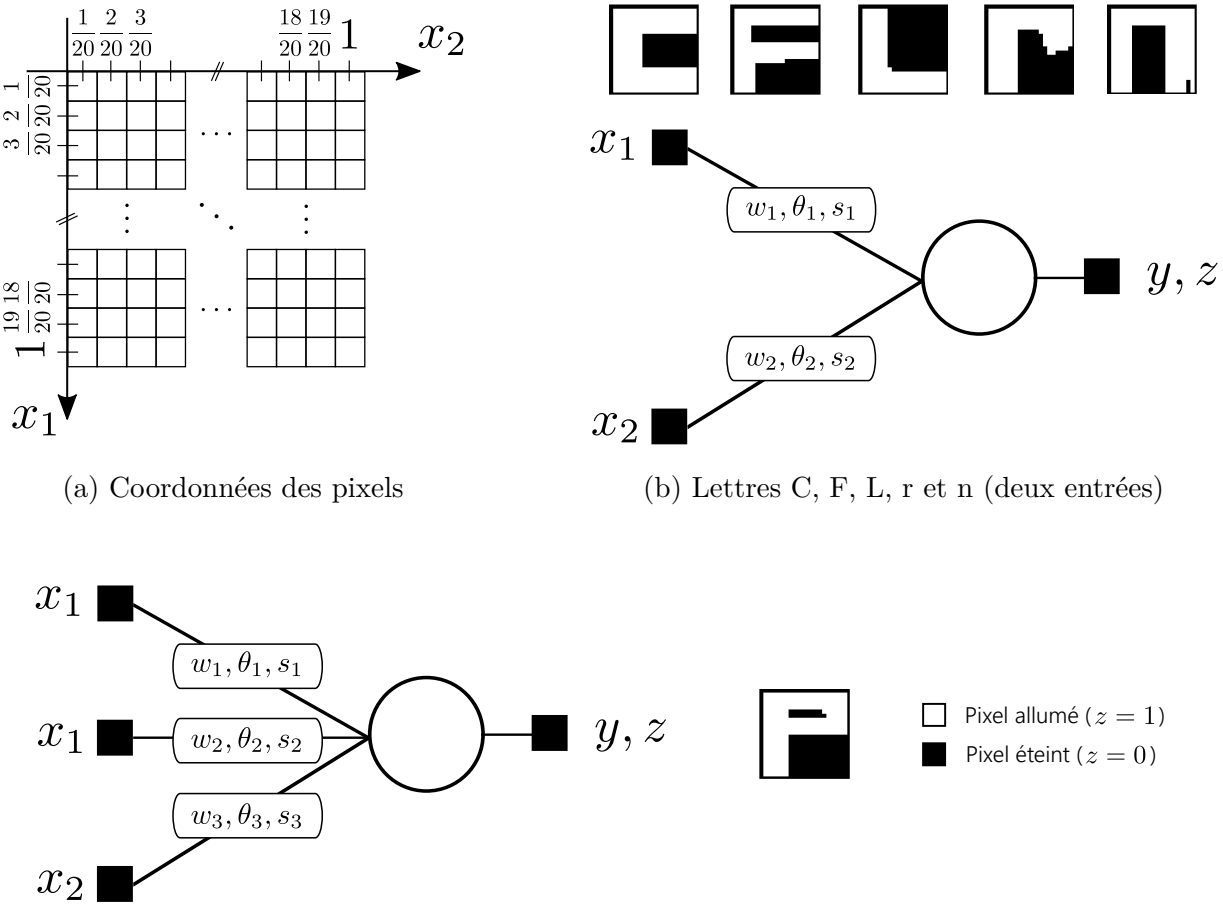
5.1 Apprentissage d'un quantron avec deux sorties cibles

On utilise des images générées par Hackenbeck-Lambert (2011) représentant les lettres C, F, L et P. Celles-ci ont été choisies parce que ce sont des images qui n'ont pas subi de traitement subséquent, c'est-à-dire qui ne résultent pas d'une série d'opérations logiques (NON, ET et OU) appliquées à plusieurs images générées par des quantrons. À ces quatres formes, on ajoute les lettres minuscules r et n générées par essai-erreur. Dans tous les cas, on a utilisé $N = 10$ PPS triangulaires par entrée et un seuil $\Gamma = 1$. Chaque image consiste en un carré de 20 pixels de côté, de sorte que l'ensemble d'entraînement contient 400 exemples. Le pixel (i, j) a pour coordonnées $(x_1, x_2) = \left(\frac{i}{20}, \frac{j}{20}\right)$ qui servent d'entrées (voir Figure 5.1a). Les lettres C, F, L, r et n utilisent ces entrées une fois chacune (Figure 5.1b) tandis que la lettre P dédouble l'entrée x_1 : le quantron correspondant travaille donc avec trois entrées (Figure 5.1c). L'Annexe B regroupe les valeurs des paramètres utilisées pour générer ces formes.

Mentionnons qu'il est normal que ces représentations de caractères alphabétiques soient légèrement imparfaites. En effet, un seul quantron a été utilisé pour les produire : des représentations plus fidèles pourraient être générées par des réseaux de plusieurs quantrons. Cette remarque s'applique particulièrement aux lettres r et n dont l'aspect esthétique n'a pas fait l'objet d'une procédure d'optimisation rigoureuse.

Précisons enfin que le problème étudié consiste à classer les *pixels* de l'image en deux classes et non pas à classer l'*image* elle-même pour déduire quel caractère alphabétique est représenté. Un tel problème de reconnaissance de caractères pourrait être abordé avec des réseaux de quantrons, mais il faudrait alors fournir en entrée du réseau les valeurs de tous les pixels de l'image pour ensuite l'entraîner à produire des sorties (premier temps d'atteinte du seuil ou

état d'activation) encodées pour représenter des caractères spécifiques [1].



(a) Coordonnées des pixels

(b) Lettres C, F, L, r et n (deux entrées)

(c) Lettre P (trois entrées, x_1 dédoublée)

Figure 5.1 Images générées par un quantron à PPS triangulaires

5.1.1 Paramètres d'apprentissage

L'apprentissage est réalisé en mode séquentiel (*online*), c'est-à-dire que les $\mathcal{N}$ pixels sont présentés en ordre aléatoire au réseau et les paramètres sont modifiés à chaque exemple. Cela permet une exploration de l'espace des paramètres dite « stochastique » qui réduit les chances de tomber dans un minimum local de la surface d'erreur (Haykin, 1999). Une époque consiste à parcourir les $\mathcal{N}$ exemples ; le processus est ainsi itéré époque après époque jusqu'à convergence ou jusqu'à ce qu'un nombre maximal d'époques soit atteint.

1. À ce sujet, voir les travaux de Pepga Bissou (2007).

Le taux d'apprentissage à une époque donnée est calculé en fonction de l'erreur à la fin de l'époque précédente. Quand une faible erreur est atteinte, on suppose être dans une configuration quasi optimale : le taux est donc réduit afin de localiser la recherche dans la région avoisinante. La forme particulière pour le taux à l'époque $n+1$ est

$$\eta_P(n+1) = \eta_P^+ + \left(\eta_P^- - \eta_P^+\right) \exp\left(-\frac{2E_z(n)}{\mathcal{N}}\right) \tag{5.1}$$

où η_P^- est le taux minimal atteint quand $E_z = 0$ et η_P^+ sert à moduler le taux maximal possible, atteint lorsque $2E_z = \mathcal{N}$. Le taux à une époque donnée dépend donc de la fraction de pixels de l'image qui sont incorrectement classés à la fin de l'époque précédente, puisque

$$\frac{2E_z}{\mathcal{N}} = \frac{1}{\mathcal{N}} \sum_{l=1}^{\mathcal{N}} \left(z^{(l)} - z_d^{(l)}\right)^2 = \frac{1}{\mathcal{N}} \sum_{l=1}^{\mathcal{N}} z^{(l)} - z_d^{(l)}.$$

La dépendance exponentielle du taux a été choisie parce qu'il s'agit d'une fonction monotone décroissante et parce qu'elle s'évalue rapidement.

Plusieurs valeurs des taux limites η_P^- et η_P^+ ont été testées pour chacun des paramètres pour arriver à la conclusion que l'ensemble de valeurs

$$\eta_w^- = \eta_s^- = 0{,}01 \quad \eta_w^+ = \eta_s^+ = 0{,}1 \quad \eta_\theta^- = 0{,}1 \quad \eta_\theta^+ = 3$$

permet d'obtenir de bons résultats sur les problèmes qui ont été étudiés.

L'évaluation des gradients locaux δ_{yy} et δ_{zy} requiert l'inverse de la dérivée temporelle de la fonction d'activation. Cette dérivée peut devenir très petite dans une situation de *hair-triggering* (voir Section 2.2.1) où la fonction d'activation franchit le seuil très près d'un maximum local. Les fortes modifications qui en découlent déstabilisent très souvent l'apprentissage, car le temps de sortie est propice à subir une discontinuité. Pour pallier cet inconvénient, on empêche ces gradients d'être supérieurs à 1 en magnitude. Plus spécifiquement, lorsqu'un gradient est supérieur à 1 en valeur absolue, on fixe simplement sa valeur à ± 1 selon son signe.

On initialise chaque poids aléatoirement suivant une loi uniforme sur $[0{,}4; 1]$ afin de favoriser l'activation des quantrons sans toutefois les saturer. On fait de même en utilisant l'intervalle $[0{,}1; 1]$ pour s, ce qui permet de couvrir une vaste gamme de rapports s/x (entre $\frac{0{,}1}{1} = 0{,}1$ et $\frac{0{,}1}{1/20} = 20$) correspondant aux entrées possibles (voir Figure 5.1a). Des lois uniformes sont

employées, car rien ne laisse supposer que certaines valeurs initiales particulières de ces paramètres pourraient produire des configurations initiales plus favorables que d'autres. On permet alors à chaque nombre réel dans l'intervalle d'être généré avec la même probabilité. Enfin, parce qu'on veut que les trains des entrées interagissent ensemble dans la configuration initiale, on fixe les délais initiaux à zéro.

Pour les étapes de l'algorithme requérant l'évaluation de fonctions sigmoïdes, il faut choisir une valeur pour le paramètre d'approximation λ introduit à l'équation (1.8). On fixe ici $\lambda = 1$, car c'est la valeur qui a produit les meilleurs résultats parmi celles qui ont été considérées (voir la Section 6.2 pour plus de détails).

Concernant la mesure de la performance, on cherche à évaluer la capacité de la méthode à repérer un ensemble de paramètres qui satisfasse à la tâche. Ainsi, lorsque l'erreur E_z n'est toujours pas nulle après 200 époques (80 000 exemples), on retient la configuration d'erreur minimale qui a été visitée.

5.1.2 Performances des heuristiques

La sélection des heuristiques optimales s'effectue en deux étapes. La première consiste à appliquer chacune des 15 heuristiques [2] individuellement pour comparer leurs performances. Par exemple, lorsque l'Heuristique 3 appliquée aux délais est testée, on pose $\partial z_i/\partial w_{ki} = 0$ pour les poids et $\partial z_i/\partial s_{ki} = 0$ pour les largeurs de noyaux (aucune heuristique n'est appliquée aux poids et aux largeurs dee noyaux). Cela revient à dire que les poids et les largeurs de noyaux sont modifiés seulement dans le but d'atteindre les temps cibles et non pas en vue de changer l'état d'activation du quantron. On procède ainsi pour tenter d'isoler la performance individuelle de chaque heuristique et pour ne pas avoir à tester la totalité des 125 combinaisons possibles [3]. Dans la seconde, on retient les deux meilleures heuristiques de chaque classe de paramètres et toutes les combinaisons possibles de ces dernières sont testées. On identifie ainsi l'agencement qui optimise les performances de la méthode.

Heuristiques individuelles

Pour chaque image, l'algorithme est exécuté 30 fois en partant de configurations initiales aléatoires afin de baser l'évaluation de sa performance sur plus d'un essai. Cette approche

2. On rappelle qu'il y a cinq heuristiques pour chaque classe de paramètres. Les trois heuristiques générales s'adaptent à chaque classe : elles génèrent donc neuf heuristiques distinctes. De plus, il y a deux heuristiques spécifiques à chaque classe, ce qui ajoute six autres heuristiques pour un grand total de 15 heuristiques.

3. Il aurait fallu plusieurs semaines de calcul ininterrompu pour tester les 125 combinaisons possibles avec le code séquentiel dans sa version actuelle. En parallélisant le code et en utilisant un serveur de calcul dédié, il serait aujourd'hui possible de réaliser cette tâche en un temps plus court.

est nécessaire dans le contexte où les résultats peuvent dépendre des valeurs initiales des paramètres des quantrons. Le nombre d'essais a été choisi en fonction de la puissance de calcul disponible : 30 essais ont permis d'appliquer la méthode sur un nombre intéressant de problèmes sans rendre le temps de calcul prohibitif. Après chaque essai, le taux de classification correcte

$$1 - \frac{2E_z}{\mathcal{N}} = 1 - \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} z^{(n)} - z_d^{(n)}$$

traduisant la proportion de pixels classés adéquatement, est calculé pour la meilleure configuration atteinte. La moyenne de cette quantité sur les 30 essais, appelé taux moyen, est finalement utilisée comme critère de performance.

Le Tableau 5.1 recueille les résultats pour les heuristiques concernant les poids. À titre indicatif, on donne aussi les performances de H0 pour laquelle aucune heuristique n'est appliquée (l'apprentissage s'effectue seulement avec y_d).

Tableau 5.1 Performances des heuristiques pour $\partial z_i / \partial w_{ki}$ (30 essais)

Heuristique	Taux moyen de classification correcte (%)						Moyenne (%)	Minimum (%)
	C	F	L	r	n	P		
H0	80,1	63,6	52,8	73,6	87,5	72,5	71,7	52,8
H1	96,7	78,5	78,1	91,4	90,9	87,6	87,2	78,1
H2	95,9	81,8	81,2	92,0	95,6	82,1	**88,1**	**81,2**
H3	96,5	81,3	87,1	93,7	94,8	92,3	**91,0**	**81,3**
H4	85,2	80,9	81,7	87,9	91,4	87,0	85,7	80,9
H5	87,3	78,2	88,5	89,9	94,4	83,4	87,0	78,2

Un premier effet observable est que toutes les heuristiques améliorent les performances de H0. En particulier, certaines permettent de passer de 52,8 % (L) et de 63,6 % (F) à plus de 80 % dans les deux cas. Pour déterminer lesquelles sont les meilleures, on utilise la moyenne et le minimum des taux de pixels correctement classés. La première donne une indication de la performance sur les six formes tandis que la seconde révèle le pire taux de classification correcte atteint. Une bonne heuristique est telle que ces deux indicateurs sont élevés. La lecture du tableau permet de conclure que H2 et H3 s'avèrent les meilleures options concernant les poids.

On poursuit similairement avec les heuristiques pour les délais dont les performances se retrouvent au Tableau 5.2. Ici, H3 et H5 produisent les moyennes les plus élevées. Par contre, elles présentent les minima les plus bas. On remarque que H4 et H1 présentent des moyennes

Tableau 5.2 Performances des heuristiques pour $\partial z_i / \partial \theta_{ki}$ (30 essais)

Heuristique	Taux moyen de classification correcte (%)						Moyenne (%)	Minimum (%)
	C	F	L	r	n	P		
H0	80,1	63,6	52,8	73,6	87,5	72,5	71,7	52,8
H1	79,1	61,5	53,9	67,6	82,5	65,1	68,3	53,9
H2	67,5	55,4	52,7	61,1	64,9	61,6	60,5	52,7
H3	83,0	67,7	51,6	71,0	87,8	65,7	**71,1**	**51,6**
H4	74,6	59,7	56,9	67,1	83,2	68,5	68,3	56,9
H5	81,4	63,5	52,5	71,6	91,5	67,3	**71,3**	**52,5**

légèrement plus faibles et des minima un peu plus hauts. Globalement, les performances de ces quatre heuristiques s'avèrent assez similaires et la sélection des meilleures dépend de l'importance relative donnée aux métriques. On choisit ici de prioriser la moyenne par rapport au minimum étant donné que la première reflète les performances générales sur toutes les formes considérées et non uniquement sur une forme particulière. Ainsi, on retient H3 et H5 pour la suite.

En comparant avec H0, il semble plus avantageux de s'abstenir d'utiliser toute heuristique étant donné que les performances de H0 sont supérieures dans la grande majorité des situations, surtout concernant la lettre P. Cela suggère que l'algorithme est capable d'assigner des délais adéquats en se basant uniquement sur les valeurs de y_d, et que les heuristiques testées se basant sur z_d viennent en quelque sorte perturber ce processus. On observe cependant que H3 et H5 apportent une certaine amélioration pour C et F, ce qui assure que malgré cette légère perte d'efficacité moyenne, il existe des cas pour lesquels ces heuristiques se révèlent pertinentes. Dans tous les cas, il serait inconcevable de conserver H0 pour la suite puisqu'elle empêche tout apprentissage des délais si seuls les états d'activation z_d sont utilisés.

Le Tableau 5.3 collige les taux moyens de classification correcte des heuristiques pour les largeurs de noyaux. On associe H1 et H4 aux meilleurs résultats sans ambiguïté. Effectivement, elles démontrent des taux de classification moyens et minimum significativement supérieurs aux autres. Celles-ci ont en commun d'utiliser la dérivée de la fonction d'activation en ξ_i, sans recourir à l'approximation de z par une sigmoïde. H4 présente un léger avantage relativement au taux minimum de classification correcte que l'on attribue à l'utilisation de l'expression analytique de l'extremum d'un train de PPS lorsqu'une seule entrée contribue à $v_i(\xi_i)$.

Tableau 5.3 Performances des heuristiques pour $\partial z_i / \partial s_{ki}$ (30 essais)

Heuristique	Taux moyen de classification correcte (%)						Moyenne (%)	Minimum (%)
	C	F	L	r	n	P		
H0	80,1	63,6	52,8	73,6	87,5	72,5	71,7	52,8
H1	95,8	78,7	87,8	86,1	98,3	78,4	**87,5**	**78,4**
H2	80,1	57,8	48,0	75,6	70,4	69,5	66,9	48,0
H3	94,1	78,3	88,2	88,4	96,3	74,1	86,6	74,1
H4	96,7	79,5	81,2	88,4	97,6	81,5	**87,5**	**79,5**
H5	96,3	79,8	83,3	88,8	98,6	75,5	87,1	75,5

Combinaison optimale d'heuristiques

Ayant identifié les deux meilleures heuristiques pour chaque classe de paramètres, le nombre de combinaisons possibles est considérablement réduit. De ce fait, on peut tester chacune des huit possibilités et retenir la meilleure par le même processus de sélection.

Tableau 5.4 Performances moyennes de combinaisons d'heuristiques (30 essais)

Combinaison H w-θ-s	Taux moyen de classification correcte (%)						Moyenne (%)	Minimum (%)
	C	F	L	r	n	P		
H 2-3-1	98,9	94,0	88,7	99,4	98,9	97,9	96,3	88,7
H 2-3-4	99,9	98,0	92,6	100,0	100,0	95,6	97,7	92,6
H 2-5-1	100,0	100,0	86,4	100,0	100,0	96,1	97,1	86,4
H 2-5-4	100,0	99,0	91,4	100,0	100,0	97,7	98,0	91,4
H 3-3-1	97,8	99,3	93,7	99,1	100,0	98,0	98,0	93,7
H 3-3-4	99,8	98,9	94,6	99,1	99,3	99,0	**98,5**	**94,6**
H 3-5-1	100,0	100,0	87,9	99,6	100,0	97,5	97,5	87,9
H 3-5-4	100,0	100,0	92,8	99,6	100,0	98,8	98,5	92,8

Immédiatement, on remarque que les huit combinaisons testées améliorent largement les performances par rapport aux heuristiques prises individuellement. En effet, le pire taux de classification parmi toutes les combinaisons et toutes les lettres est de 88,7 % et survient pour la lettre L (voir Tableau 5.4). Un tel taux est de l'ordre des meilleurs résultats atteints par les heuristiques individuelles pour w et s. Également, les taux moyens et minimum dépassent 95 % et 88 % respectivement pour toutes les combinaisons, signifiant qu'elles accomplissent toutes les tâches d'apprentissage de manière satisfaisante.

On désire quand même identifier la meilleure combinaison afin de présenter un algorithme final. On constate que l'Heuristique 3 pour les poids s'avère plus efficace que l'Heuristique 2 : à heuristiques égales pour θ et s, les combinaisons avec H3 montrent toujours des taux supérieurs. De la même façon, on observe que l'Heuristique 4 pour les largeurs de noyaux apporte systématiquement une amélioration par rapport à l'Heuristique 1. Il reste alors à choisir parmi H 3-3-4 et H 3-5-4. Malgré des moyennes égales, la première possède un taux minimum plus élevé : H 3-3-4 est donc la combinaison optimale suite aux expériences menées.

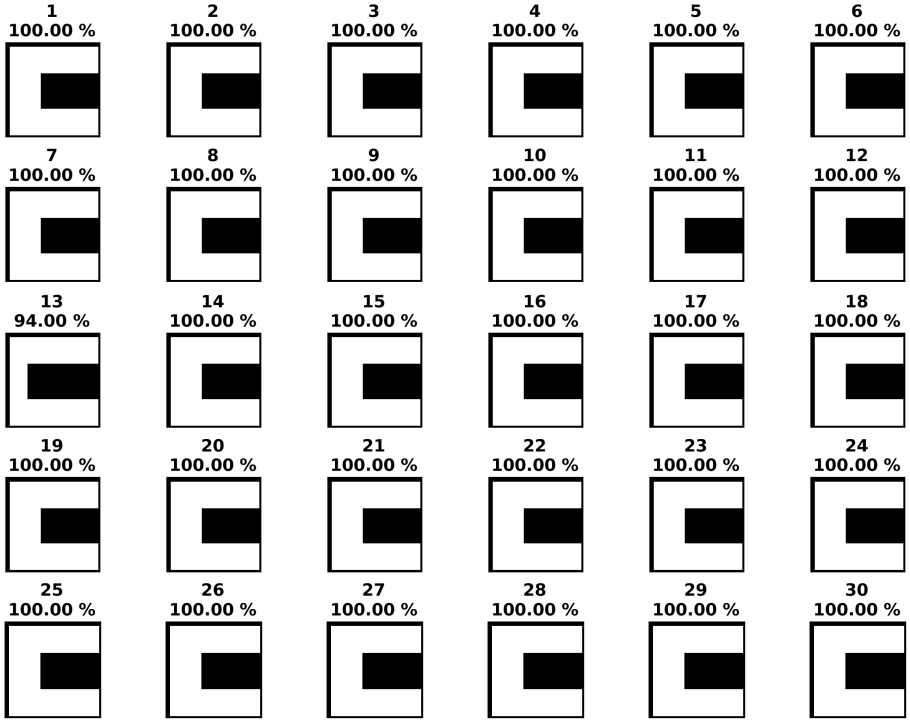
Images obtenues suite à l'apprentissage

Il est intéressant d'observer les images optimales obtenues suite à chacun des 30 essais pour chaque lettre pour plusieurs raisons. Premièrement, les taux de classification correcte présentés sont des moyennes cachant une certaine variabilité pouvant être observée sur les mosaïques d'images. Deuxièmement, cela permet d'associer aux taux de classification une certaine qualité visuelle de la représentation de la lettre. Dernièrement, on peut reconnaître quelles parties de l'image sont les plus difficiles à apprendre en analysant les essais n'ayant pas convergé.

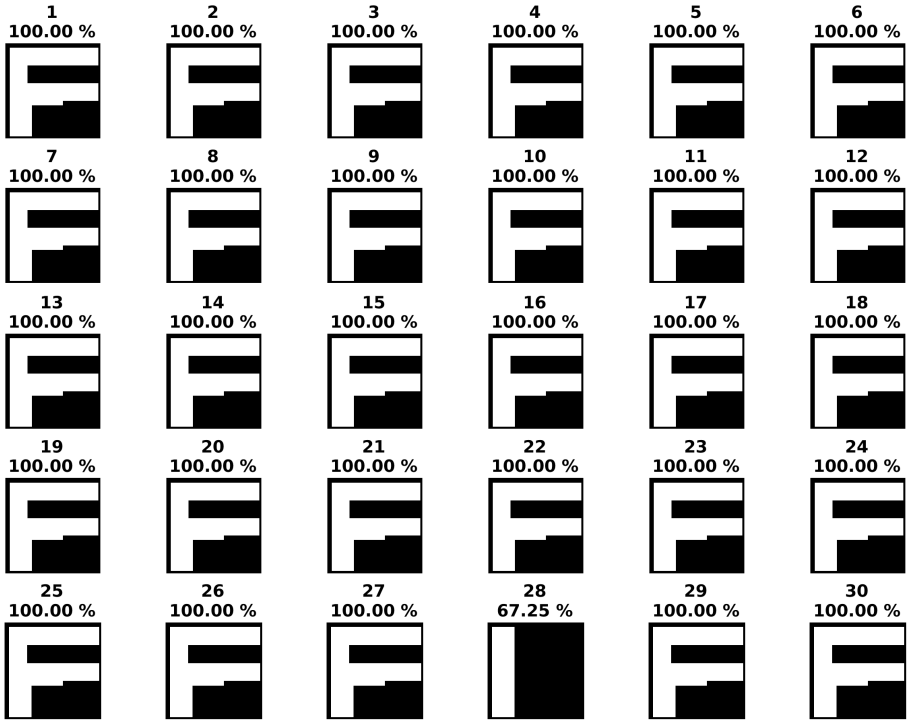
Afin de bien saisir la signification du taux de classification correcte, il faut être conscient qu'il descend rarement sous la barre des 50 %. Parmi les deux configurations dégénérées d'un quantron qui est actif pour tous les pixels et d'un quantron qui reste silencieux pour tous les pixels, l'erreur dans un des deux cas est toujours supérieure ou égale à 50 % selon la proportion de pixels allumés dans l'image cible. En fait, une erreur nulle voudrait dire que le quantron est parvenu à reproduire le négatif de l'image, qui s'avère généralement aussi complexe à apprendre que l'original. Ainsi, un taux situé entre 40 et 60 % correspond le plus souvent à une image presqu'unie résultant d'une divergence de l'algorithme.

L'apprentissage de la lettre C est un franc succès : seul l'essai 13 n'a pas reproduit la forme parfaitement. Pour celui-ci, 94 % des pixels sont correctement classés et la forme obtenue s'assimile facilement à la lettre étant donné que l'erreur résulte en un simple amincissement de la barre verticale du caractère (Figure 5.2a).

Pour la lettre F de la Figure 5.2b, on associe au seul essai qui n'a pas convergé totalement (essai 28) une erreur résiduelle de 67,25 %. La barre verticale obtenue ne représente pas adéquatement la lettre en l'absence des deux barres horizontales. Cette barre est typique d'un certain cas de divergence où un des poids atteint presque le seuil et où l'autre poids est presque nul en magnitude de sorte que les deux trains n'interagissent presque plus.

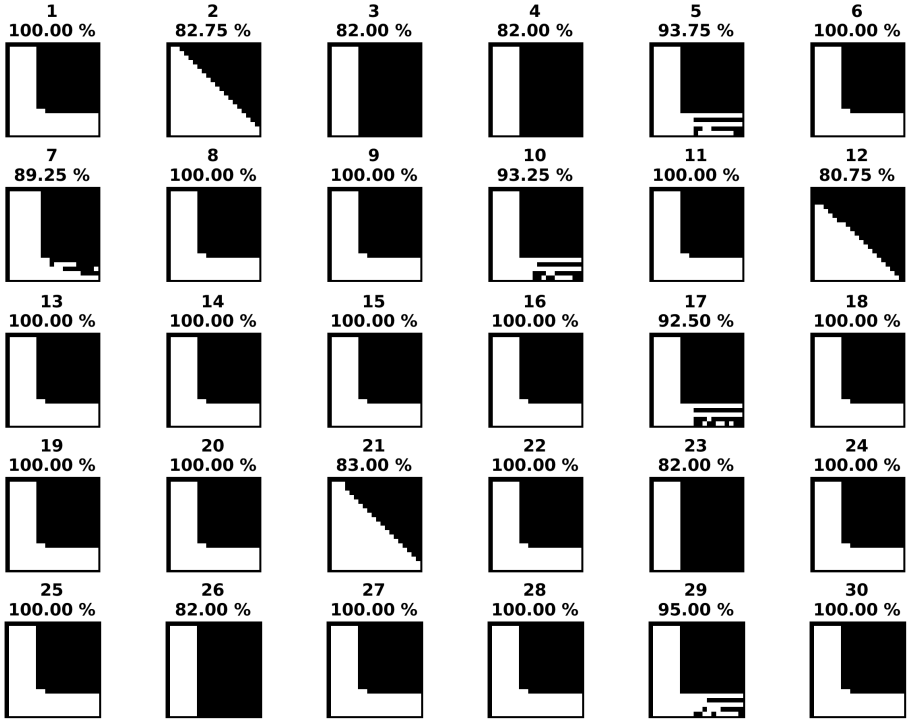


(a) Lettre C

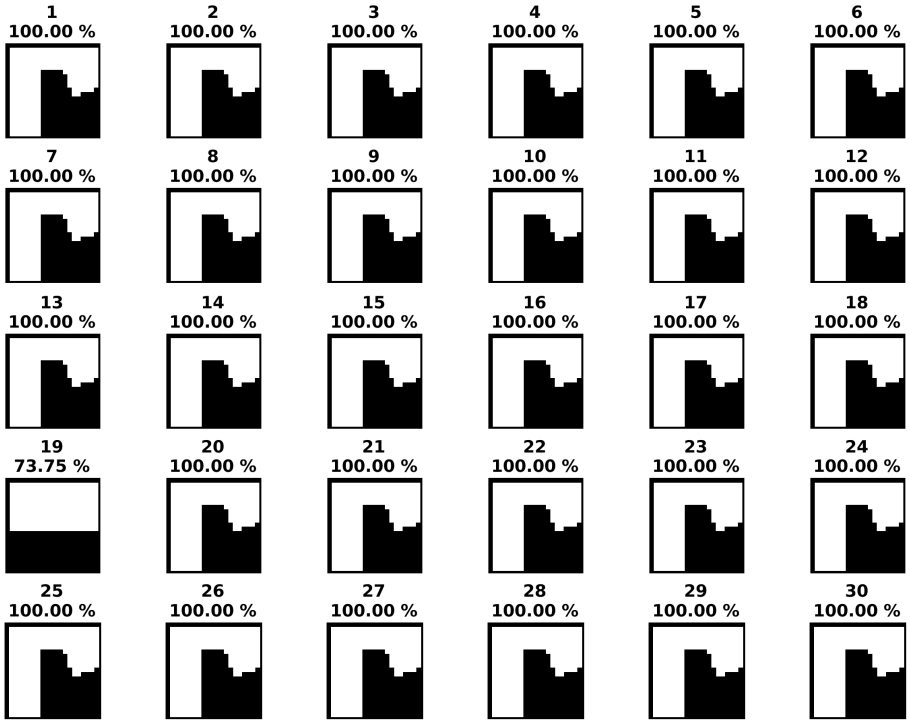


(b) Lettre F

Figure 5.2 Mosaïques des images résultant de l'apprentissage avec H 3-3-4 (1 sur 3)

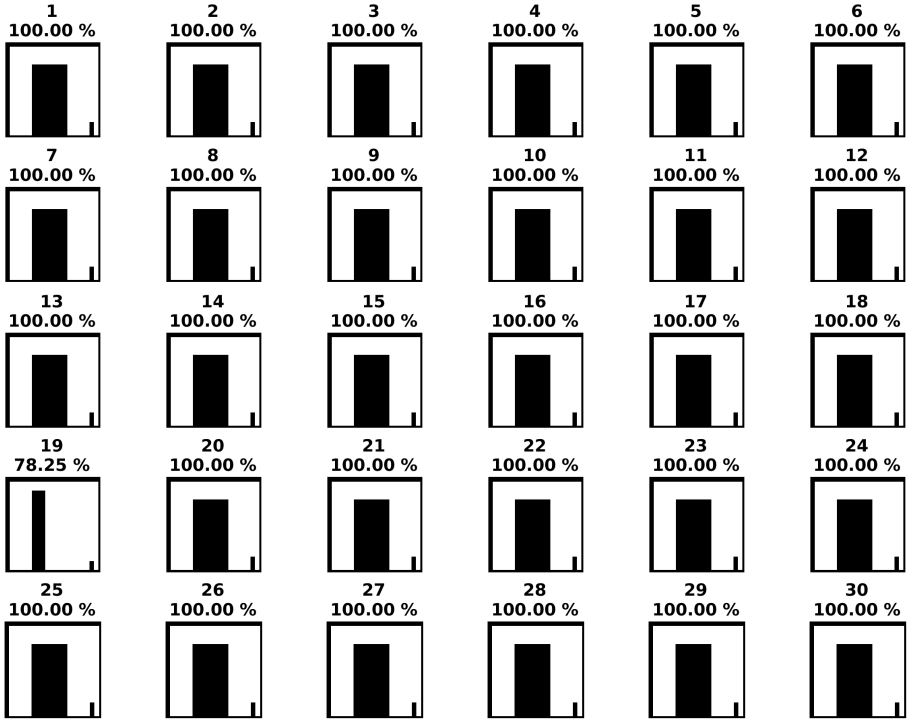


(c) Lettre L

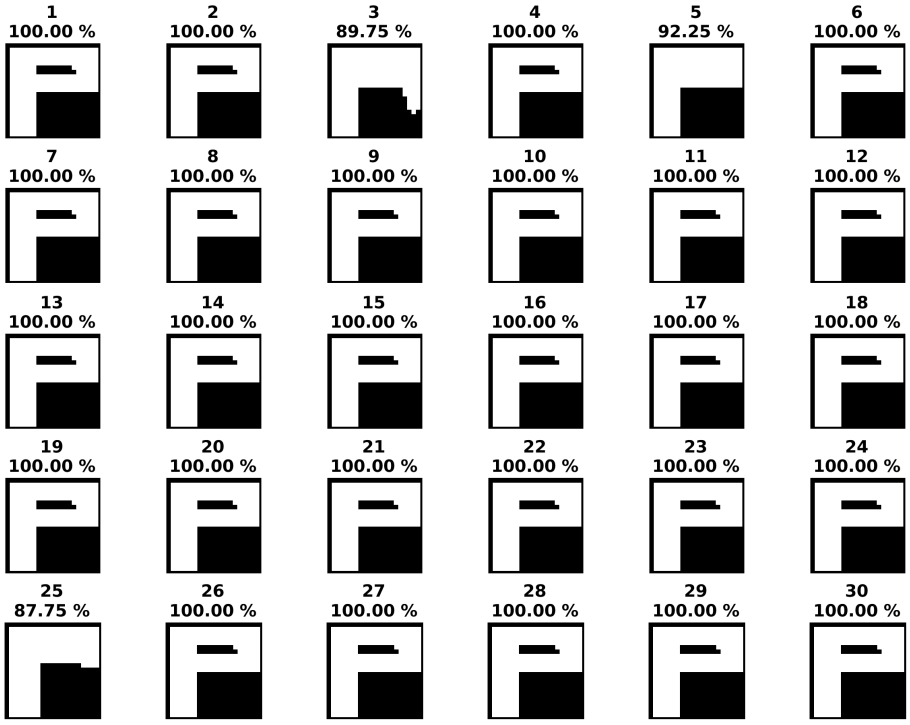


(d) Lettre r

Figure 5.2 Mosaïques des images résultant de l'apprentissage avec H 3-3-4 (2 sur 3)



(e) Lettre n



(f) Lettre P

Figure 5.2 Mosaïques des images résultant de l'apprentissage avec H 3-3-4 (3 sur 3)

Il s'avère que l'algorithme peine davantage à apprendre la lettre L (5.2c) étant donné que 12 des 30 essais n'ont pas convergé entièrement. Les essais 3, 4, 23 et 26 ont mené à des barres verticales semblables à celles de la lettre F. Trois essais (2, 12 et 21) ont résulté en une région triangulaire inférieure. Cette dernière est produite lorsque w_1 est légèrement supérieur au seuil, que w_2 est négatif et que $\theta_2 < \theta_1$ tout en permettant une interaction entre les trains. En l'absence du train 2, l'image serait saturée, mais sa présence permet de diminuer l'amplitude du train 1 et de créer la région éteinte. Malgré le fait que les images ainsi obtenues ne ressemblent pas à l'image cible, les taux de classification franchissent les 80 %.

Enfin, on reconnaît des variantes bruitées de la lettre L pour cinq essais (5, 7, 10, 17 et 29). Pour celles-ci, l'algorithme ne parvient pas à ajuster correctement les fines interactions requises pour produire la patte du bas. En l'absence du train 1, la barre horizontale ne pourrait pas exister et on verrait seulement la barre verticale produite par le train 2. Quand x_1 est faible, le train 1 est court et puisque $\theta_1 < \theta_2$, il se termine avant que le train 2 commence. Lorsque x_1 augmente, l'étendue temporelle du train 1 s'accroît et son extrémité finale interagit avec le train 2 pour activer le quantron. C'est cette dynamique que l'algorithme n'arrive pas à reproduire complètement. Les taux de classification de ces essais sont de l'ordre de 90 à 95 %. En comparant avec les autres cas de divergence, on conclut que cette métrique n'est pas réellement proportionnelle au degré de ressemblance avec l'image cible. En effet, la presque totalité des pixels doit être correctement classée afin de pouvoir reconnaître visuellement la lettre.

L'apprentissage de la lettre r s'avère très satisfaisant avec un seul essai imparfait. À l'essai 19, l'algorithme a produit une large bande horizontale en tentant de former le haut de la lettre. Il s'agit encore une fois d'un cas de divergence où une seule entrée participe à l'activation du quantron.

Le cas de la lettre n est similaire. L'unique tentative ayant échoué a quand même réussi à générer une forme similaire malgré un taux de classification inférieur à 80 %. Sur l'image 19, les pixels mal classés forment un bloc adjacent à la patte droite issu d'une valeur un peu trop grande pour θ_2 amenant l'entrée 2 à interagir trop tôt avec l'entrée 1, toutes deux excitatrices.

Une forte proportion (90 %) des essais a convergé sans bavure en ce qui a trait à la lettre P. Les trois tentatives restantes (3, 5 et 25) ont produit des formes comparables au caractère alphabétique auxquelles il manque cependant l'îlot supérieur. La troisième entrée (x_2) sert ici à produire la barre verticale tandis que les deux autres entrées (x_1) interfèrent ensemble

pour former l'îlot noir. Le premier train débute bien avant les deux autres et interagit avec eux seulement lorsque x_2 est suffisamment grand afin de produire la courte barre verticale du coin supérieur droit de l'image. Étant conscient du mécanisme de formation de la lettre, on conclut que la principale difficulté que rencontre l'algorithme réside à reproduire l'interférence complexe générant l'îlot.

Les résultats présentés mènent à conclure que l'algorithme parvient à entraîner le quantron avec succès en connaissant y_d et z_d. Suite à cela, nous avons tenté d'apprendre les mêmes caractères alphabétiques en utilisant z_d uniquement, puisqu'il s'agit de la seule information réellement disponible pour reconnaître une image fondamentalement binaire. La Figure 5.3a montre cependant que les performances subissent une importante détérioration alors que tous les taux moyens de classification sont inférieurs à 85 %. Pour retrouver des performances comparables à celles du Tableau 5.4, il faut utiliser des réseaux à 10 neurones cachés. La Figure 5.3b montre l'évolution de la meilleure image et de la pire image obtenues lorsque le nombre de neurones cachés est augmenté. On constate qu'avec 10 neurones cachés, les taux de classification des meilleures images produites par l'algorithme frôlent ou atteignent 100 %, ce qui n'est pas le cas avec un quantron seul. De plus, les taux de classification des pires images s'améliorent significativement en augmentant la taille du réseau, même s'il reste parfois difficile de reconnaître les lettres qu'elles sont censées représenter.

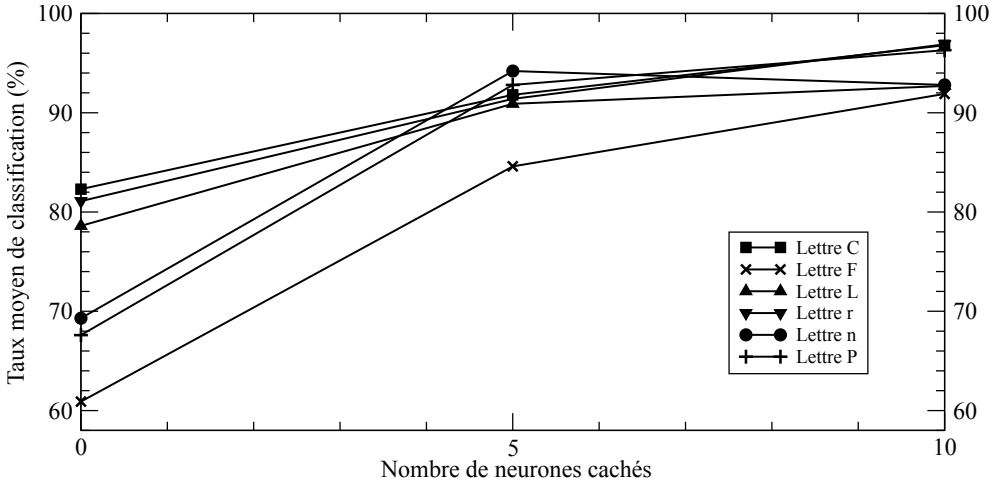
Ces résultats démontrent d'une part le rôle important que joue un ensemble cohérent de valeurs cibles y_d lors de l'apprentissage, et d'autre part que l'algorithme doit parfois employer des architectures de complexité non minimale pour résoudre certaines tâches étant donné qu'il ne parvient pas toujours à converger vers une erreur de classification nulle même lorsqu'une solution existe.

On poursuit en s'intéressant au comportement de la méthode dans ce nouveau contexte en étudiant de nouvelles images binaires composées d'un plus petit nombre de pixels. Cela ne signifie pas que les formes considérées soient plus simples pour autant, puisque la majorité d'entre elles présentent des frontières hautement non linéaires.

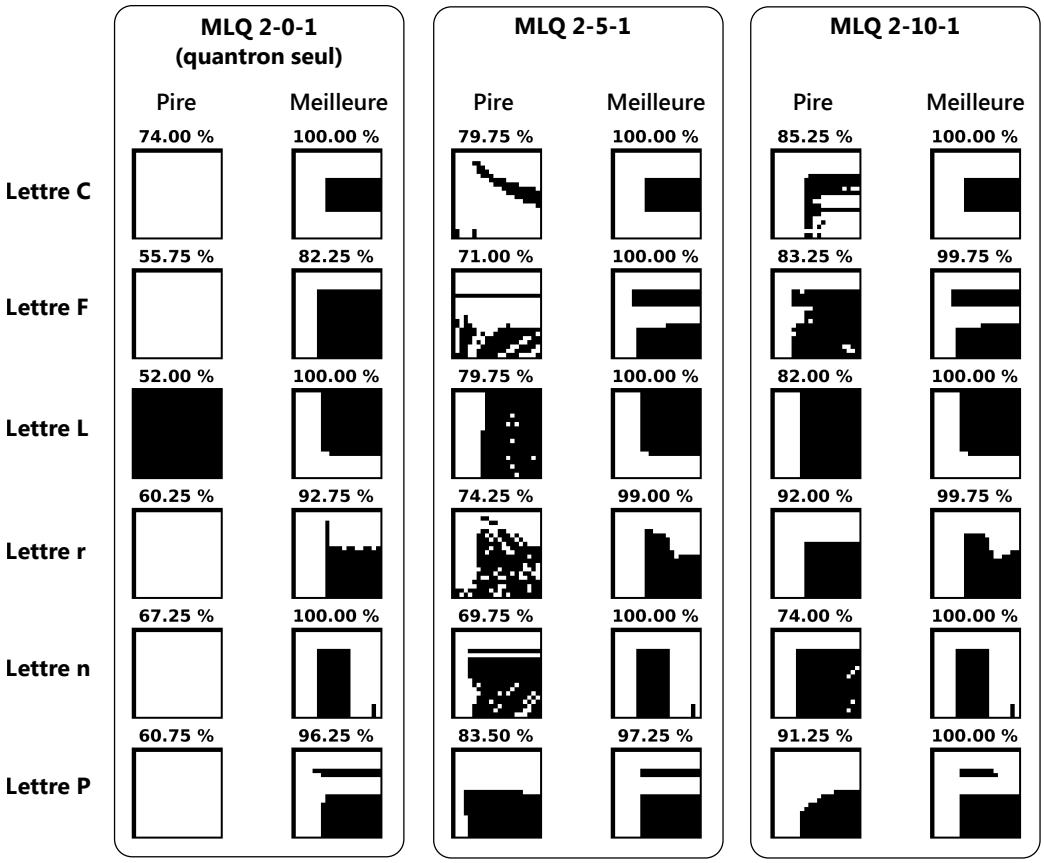
5.2 Apprentissage d'un MLQ avec les états d'activation cibles

Muni de la combinaison d'heuristiques optimale H 3-3-4, on entraîne des MLQ à deux entrées et une couche cachée à résoudre les problèmes de classification illustrés à la Figure 5.4 [4]. Ceux-ci n'ont pas été préalablement générés par un MLQ, de sorte que seules les valeurs de

4. Il faut porter une attention au système d'axes qui prend ici une forme conventionnelle.



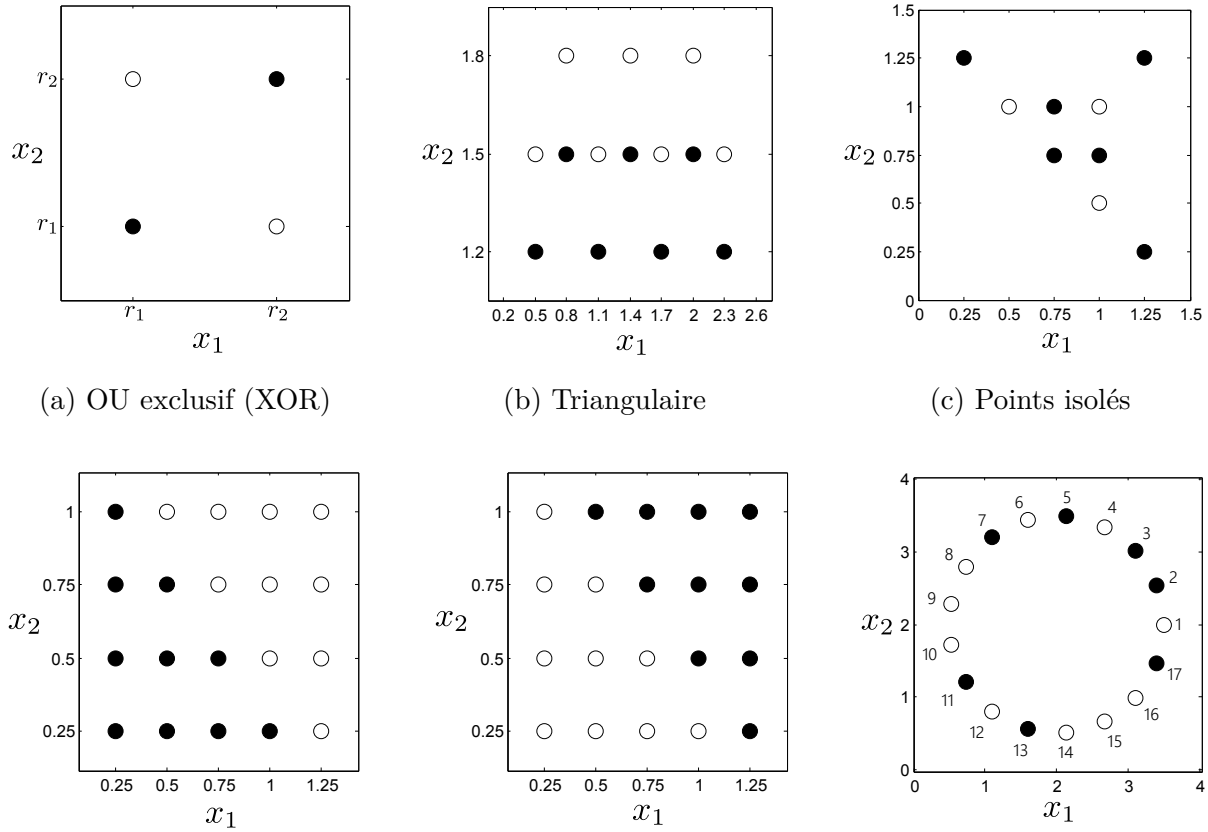
(a) Taux moyen de classification



(b) Meilleures et pires images parmi 30 essais

Figure 5.3 Performance de l'algorithme avec heuristiques optimales sur les tâches de classification de pixels de caractères alphabétiques en utilisant z_d uniquement pour des réseaux 2-0-1 (quantrons seuls), 2-5-1 et 2-10-1

z_d sont à la disposition de l'algorithme. Puisqu'on ne sait pas si un MLQ peut produire de telles frontières *a priori*, on augmente progressivement le nombre de neurones dans la couche cachée de 0 (absence de couche cachée) à 10 afin d'allouer au modèle une flexibilité suffisante correspondant à la complexité de la tâche.



(a) OU exclusif (XOR) (b) Triangulaire (c) Points isolés

(d) Linéaire 1 (e) Linéaire 2 (f) Heptadécagone

Figure 5.4 Problèmes pour l'apprentissage avec z_d (noir : silencieux, blanc : actif)

Le premier problème est celui du XOR, un banc d'essai répandu pour les problèmes non linéairement séparables. Ayant défini deux réels positifs $r_1 < r_2$, l'ensemble d'entraînement est formé des quatre couples qu'on peut former avec ceux-ci. Le MLQ doit être actif si $x_1 \neq x_2$ et rester silencieux autrement. Pour les tests, on utilise $(r_1, r_2) = (0{,}3; 0{,}7)$ et $(r_1, r_2) = (1, 2)$ pour observer l'influence du couple de réels choisis sur les performances. Les deux problèmes suivants ont été proposés par de Montigny (2014). Ils reproduisent une frontière triangulaire et un ensemble de points isolés tous deux non linéairement séparables. On poursuit avec deux problèmes linéairement séparables. Chacun est le complémentaire de l'autre afin d'étudier l'impact de l'attribution des états d'activation aux deux classes. Enfin, on propose le problème original suivant. On considère les sommets d'un heptadécagone régulier numérotés de 1 à 17

en sens antihoraire à partir du point le plus à droite. Si le nombre associé au sommet est premier, le quantron doit rester éteint, tandis que si ce nombre est composé, il doit être actif.

5.2.1 Paramètres d'apprentissage

Les paramètres d'apprentissage restent pour la majorité inchangés. Une différence apparaît au niveau du taux d'apprentissage qui ne dépend plus de l'erreur, mais qui décroît simplement avec le nombre n d'itérations.

$$\eta_P(n) = \eta_P^+ + \left(\eta_P^- - \eta_P^+\right) \exp\left(-\frac{n}{\tau_P}\right) \tag{5.2}$$

On procède ainsi parce que les ensembles d'entraînement considérés sont substantiellement plus petits que pour les caractères alphabétiques. L'erreur pouvant seulement prendre un petit nombre de valeurs distinctes, le taux n'aurait pas varié beaucoup avec la première méthode. Avec la nouvelle procédure, on choisit

$$\eta_w^- = \eta_s^- = 0{,}01 \quad \eta_w^+ = \eta_s^+ = 0{,}1 \quad \eta_\theta^- = 0{,}1 \quad \eta_\theta^+ = 1 \quad \tau_w = \tau_\theta = \tau_s = 5000.$$

On permet à l'algorithme de parcourir un maximum de 10 000 époques.

5.2.2 Résultats

Cette section regroupe les résultats des expériences menées. Pour chaque problème et chaque architecture de réseau, on indique le nombre d'essais ayant convergé parmi les 30 tentatives et le taux moyen de classification correcte. Afin de comprendre ce que ces quantités représentent, on présente un bref exemple de calcul. Admettons que l'algorithme soit exécuté cinq fois sur un problème contenant $\mathcal{N} = 10$ points à classer et qu'on obtienne respectivement 10, 10, 6, 10 et 9 points correctement classés à la fin de chacun des cinq essais. Dans ce cas fictif, il y aurait trois essais convergents où tous les points auraient été bien classés tandis que le taux moyen de classification correcte sur les cinq essais serait donné par

$$\frac{1}{5}\left(\frac{10}{10} + \frac{10}{10} + \frac{6}{10} + \frac{10}{10} + \frac{9}{10}\right) = \frac{45}{50} = 90{,}0\,\%.$$

Si tous les essais ont convergé, le taux moyen de classification est nécessairement égal à 100 %. Notons qu'un taux moyen de classification élevé peut être obtenu même si certains essais n'ont pas permis d'identifier une configuration où tous les points sont correctement classés.

Les performances sur le XOR (0,3; 0,7) sont très satisfaisantes. Le Tableau 5.5 indique que l'algorithme parvient à entraîner un seul quantron à apprentre le XOR pour plus de 50 %

Tableau 5.5 Performances sur le problème du XOR (0,3; 0,7) (30 essais)

	Nombre de neurones cachés							
	0	1	2	3	4	5	8	10
Essais convergents (/30)	17	3	22	28	28	29	30	30
Taux moyen de classification (%)	87,5	70,0	93,3	98,3	98,3	99,2	100,0	100,0

Tableau 5.6 Performances sur le problème du XOR (1, 2) (30 essais)

	Nombre de neurones cachés							
	0	1	2	3	4	5	8	10
Essais convergents (/30)	0	0	6	15	16	22	28	28
Taux moyen de classification (%)	67,5	69,2	79,2	87,5	87,5	93,3	98,3	98,3

des essais. On observe ensuite une nette diminution de la performance pour le MLQ 2-1-1, alors qu'elle augmente significativement pour le MLQ 2-2-1 et continue de croître plus il y a de neurones cachés. À partir de trois neurones cachés, presque tous les essais convergent et le taux moyen de classification correcte frise ou atteint 100 %.

L'histoire est différente pour le XOR (1,2) pour lequel la méthode n'arrive pas à entraîner un neurone seul, pas plus qu'un réseau 2-1-1 (voir Tableau 5.6). Il ne faut pas en déduire qu'il est impossible pour un seul quantron de le résoudre : il est fort possible que ce soit plutôt l'algorithme qui n'explore pas suffisamment l'espace des configurations pour tomber sur un ensemble de paramètres adéquats. Cela dit, en ajoutant des neurones cachés, les performances s'améliorent et plus de 90 % des tentatives convergent avec huit et dix neurones.

Les problèmes de la frontière triangulaire et des points isolés s'avèrent plus difficiles à résoudre. Les deux restent irrésolus par un quantron seul et le MLQ 2-1-1, alors que le réseau 2-2-1 réussit très rarement à accomplir la tâche. Il y a une certaine stagnation de la performance entre 3 et 5 (triangulaire) et 3 et 8 (points isolés) neurones cachés où le taux moyen de classification augmente timidement. Les meilleurs résultats surviennent pour le MLQ 2-10-1 qui résout parfaitement les deux problèmes pour les deux tiers des tentatives environ. Il ne s'agit pas d'une proportion spécialement élevée, mais les taux moyens de classification supérieurs à 95 % indiquent que lorsque la convergence n'est pas atteinte, l'algorithme parvient

tout de même à classer correctement la grande majorité des points comme la Figure 5.5 permet de s'en convaincre.

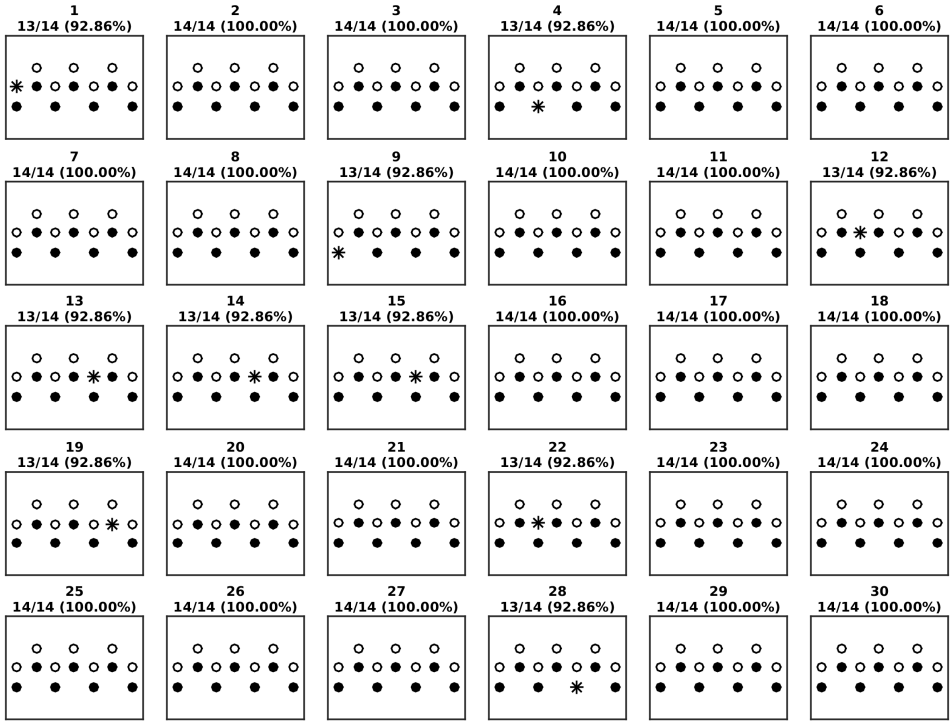
Tableau 5.7 Performances sur le problème de la frontière triangulaire (30 essais)

	Nombre de neurones cachés							
	0	1	2	3	4	5	8	10
Essais convergents (/30)	0	0	1	8	7	9	15	20
Taux moyen de classification (%)	84,1	76,0	84,3	91,2	91,7	93,3	96,0	97,6

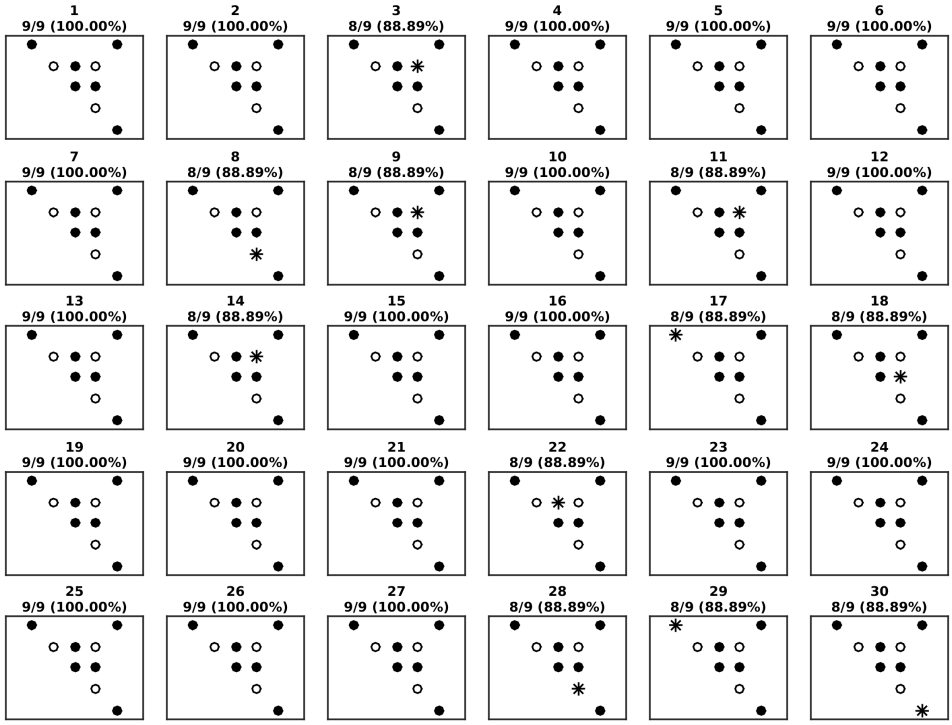
Tableau 5.8 Performances sur le problème des points isolés (30 essais)

	Nombre de neurones cachés							
	0	1	2	3	4	5	8	10
Essais convergents (/30)	0	0	2	8	7	9	7	19
Taux moyen de classification (%)	66,7	66,7	86,7	89,6	90,7	91,9	91,1	95,9

Les points mal classés y sont représentés par des astérisques ($*$) et on remarque qu'un seul pixel est incorrectement reproduit pour toutes les instances n'ayant pas convergé (triangulaire : 1, 4, 9, 12 à 15, 19, 22 et 28 ; points isolés : 3, 8, 9, 11, 14, 17, 18, 22 et 28 à 30). Pour les deux tâches, ce pixel varie d'essai en essai et on ne peut associer une région particulière des images aux difficultés de l'algorithme à converger entièrement.



(a) Frontière triangulaire (2-10-1)



(b) Points isolés (2-10-1)

Figure 5.5 Mosaïques résultant de l'apprentissage (triangulaire et points isolés)

Toutes les architectures de MLQ peinent à résoudre complètement le premier problème linéaire, bien que l'on observe que le taux moyen de classification augmente avec le nombre de neurones cachés. Pourtant, la méthode proposée parvient à résoudre parfaitement le second problème linéaire avec un haut taux de réussite lorsqu'il y a plus de cinq neurones cachés. Cela démontre qu'un problème et son complémentaire peuvent présenter deux niveaux de complexité bien différents pour un MLQ.

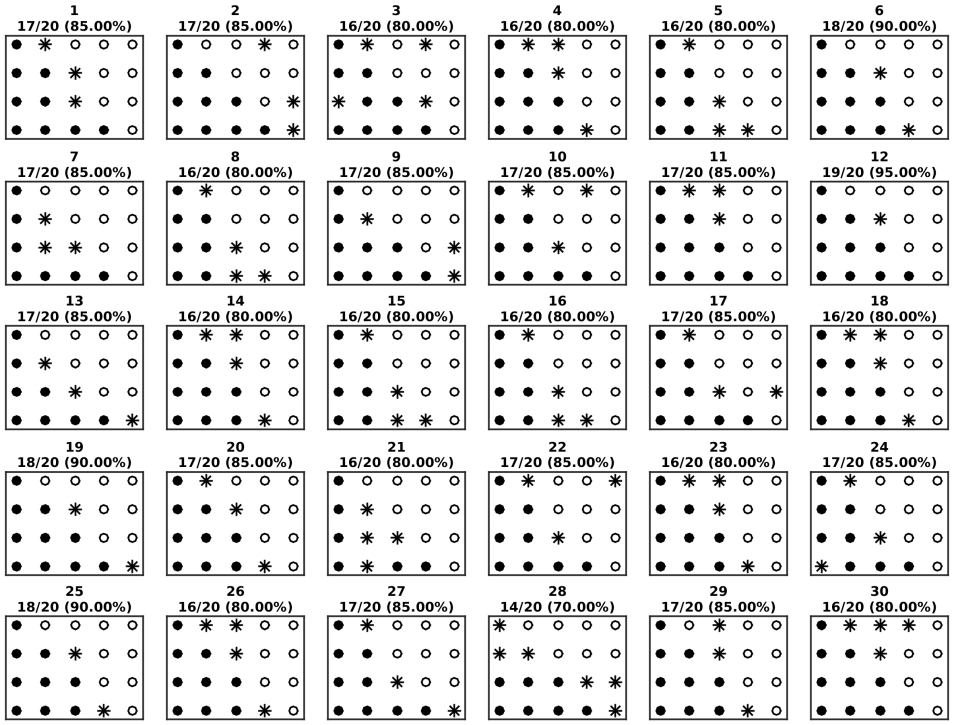
Tableau 5.9 Performances sur le problème à frontière linéaire 1 (30 essais)

	Nombre de neurones cachés							
	0	1	2	3	4	5	8	10
Essais convergents (/30)	0	0	0	0	1	0	0	0
Taux moyen de classification (%)	68,3	55,3	72,5	79,5	83,0	83,3	84,2	84,0

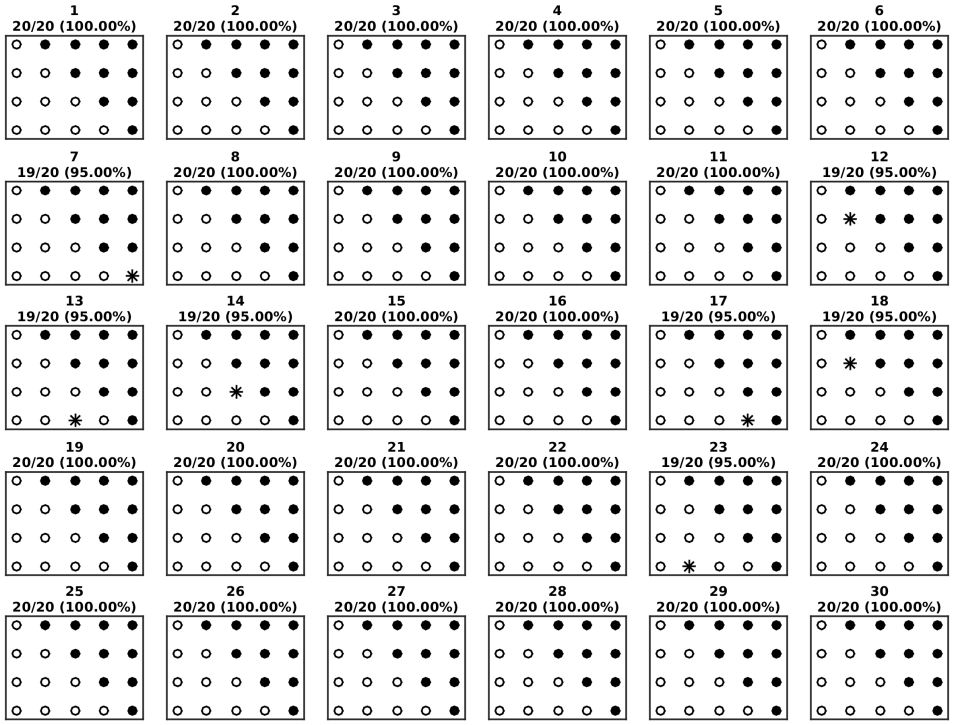
Tableau 5.10 Performances sur le problème à frontière linéaire 2 (30 essais)

	Nombre de neurones cachés							
	0	1	2	3	4	5	8	10
Essais convergents (/30)	0	0	11	13	22	23	28	30
Taux moyen de classification (%)	84,2	95,0	96,8	97,2	98,5	98,8	99,7	100,0

Dans le cas présent, on peut attribuer ce phénomène aux zones de surexcitation identifiées par Lastère (2005) et que l'on peut expliquer directement grâce au Théorème 2 pour des PPS triangulaires. Pour une entrée donnée avec w, s et N fixés, on peut montrer sans difficulté que $\max \nu(t)$ est une fonction décroissante de x en analysant (3.11). Si $wN > \Gamma$, il existe alors une valeur critique x_c en-deçà de laquelle le train de PPS atteint toujours le seuil par lui-même. Dans la situation où il n'existe pas un autre train inhibiteur pour empêcher cette entrée d'activer le quantron, une bande de surexcitation semi-infinie $0 < x < x_c$ apparaît donc. Chaque entrée possédant une bande, le quantron tend à s'activer pour les pixels près des axes de référence. La différence entre les deux problèmes à frontières linéaires réside là : le premier va à l'encontre des bandes de surexcitation en incitant le quantron à être silencieux pour de faibles valeurs de x_1 et x_2 tandis que le second problème s'harmonise mieux avec cette particularité du quantron en l'appelant à être actif dans ces régions.



(a) Frontière linéaire 1 (2-5-1)



(b) Frontière linéaire 2 (2-5-1)

Figure 5.6 Mosaïques résultant de l'apprentissage (frontières linéaires)

On a affirmé qu'un quantron possède une puissance de calcul supérieure à celle d'un perceptron, car le premier peut résoudre le problème du XOR non linéairement séparable alors que le second en est incapable. On a pourtant soumis au quantron deux problèmes linéairement séparables aisément résolubles par un seul perceptron et plusieurs neurones cachés ont dû être employés pour obtenir des résultats satisfaisants. Ce phénomène n'est pas nouveau, puisque Lastère (2005) a également étudié un problème linéairement séparable avec son algorithme jumelé à un MLQ 2-8-1 et a observé que le réseau ne parvenait pas à le résoudre complètement. Est-ce dire qu'un seul quantron ne peut résoudre un problème linéairement séparable quelconque ?

Encore une fois, la méthode d'entraînement proposée ici n'est certainement pas infaillible et il se peut qu'un meilleur algorithme parvienne éventuellement à trouver une configuration du réseau qui résolve n'importe quel problème linéairement séparable. Ceci étant dit, l'analyse des bandes de surexcitation à l'aide du Théorème 2 révèle qu'un quantron à une seule entrée avec $wN > \Gamma$ agit comme une porte à seuil (*threshold gate*) : si $0 < x \leq x_c$, le quantron est actif alors que si $x > x_c$, il reste silencieux. La Figure 5.7, où l'on a tracé le maximum de la fonction d'activation d'un quantron à une seule entrée en fonction de x pour s fixé, illustre ce comportement.

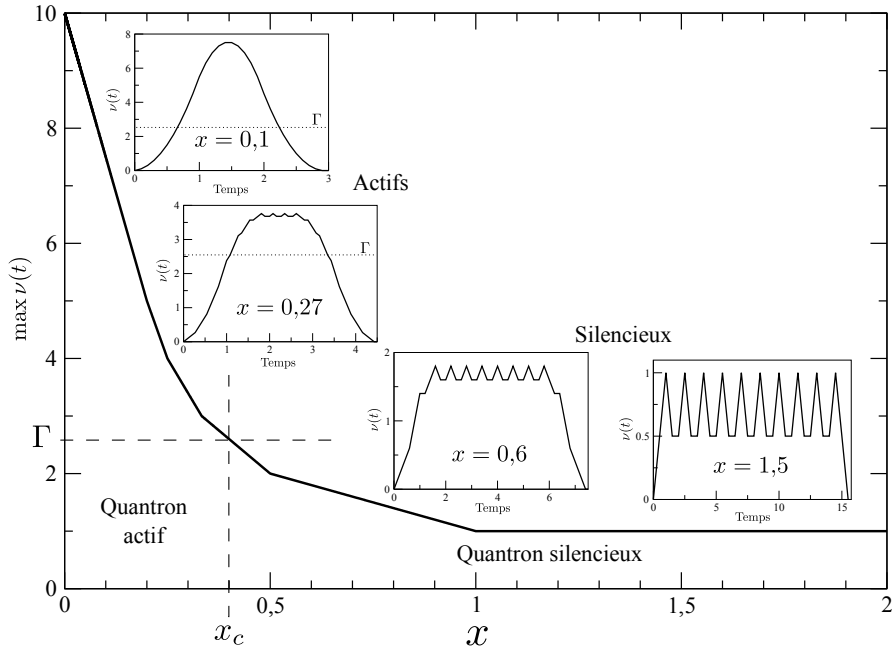


Figure 5.7 Maximum de la fonction d'activation d'un quantron à une entrée en fonction de x ($w = 1$, $s = 1$ et $N = 10$).

Ainsi, un seul quantron à une entrée pourrait résoudre n'importe quel problème linéairement séparable si on définissait cette entrée $x = c_0 + c_1x_1 + c_2x_2$ comme une combinaison linéaire des coordonnées des points du problème. Il faudrait alors ajuster les coefficients c_i par une méthode de descente du gradient utilisant (4.41) afin de minimiser l'erreur. Dans ce cas, le quantron ne serait pas véritablement entraîné étant donné que ses paramètres w et s seraient fixés et que l'algorithme d'apprentissage viserait davantage à optimiser la procédure de prétraitement des données. Cependant, cela montre que la complexité d'un problème dépend invariablement de la façon dont l'information est présentée au réseau.

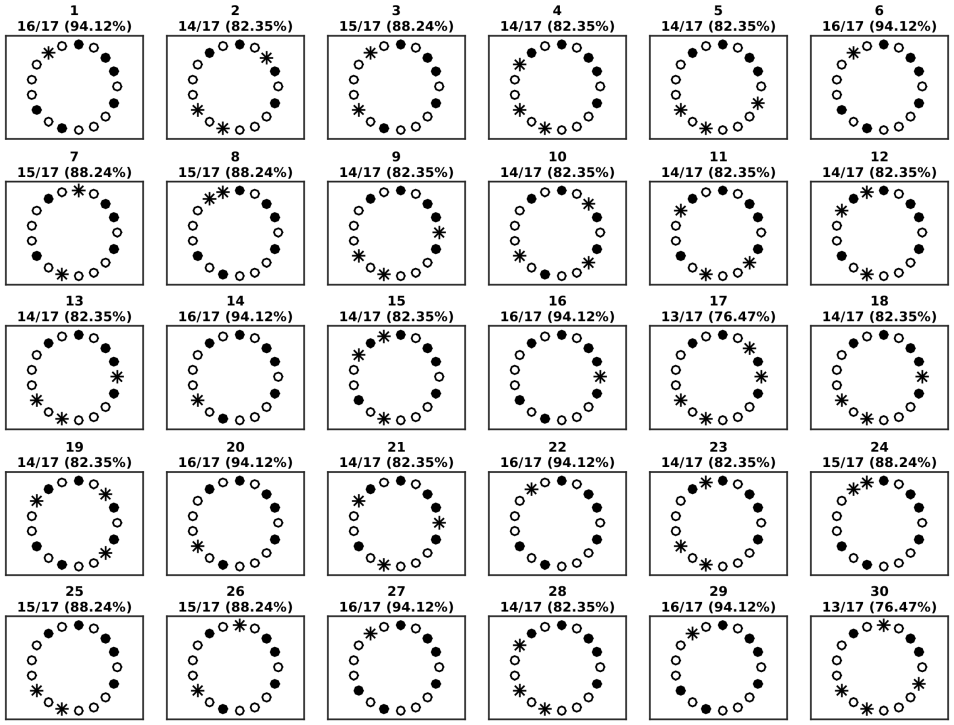
On complète avec les résultats du problème de l'heptadécagone regroupés au Tableau 5.11. L'idée derrière ce problème était de voir comment le quantron réagirait à un ensemble de points ayant une certaine symétrie circulaire (en négligeant l'appartenance aux classes) peu présente dans les tâches abordées jusqu'à présent. Pour l'attribution des classes, on a voulu utiliser une procédure qui ne permette pas d'associer facilement les coordonnées d'un point à sa classe, d'où le recours à la numérotation des points et à la propriété d'un nombre d'être premier ou composé.

Tableau 5.11 Performances sur le problème de l'heptadécagone (30 essais)

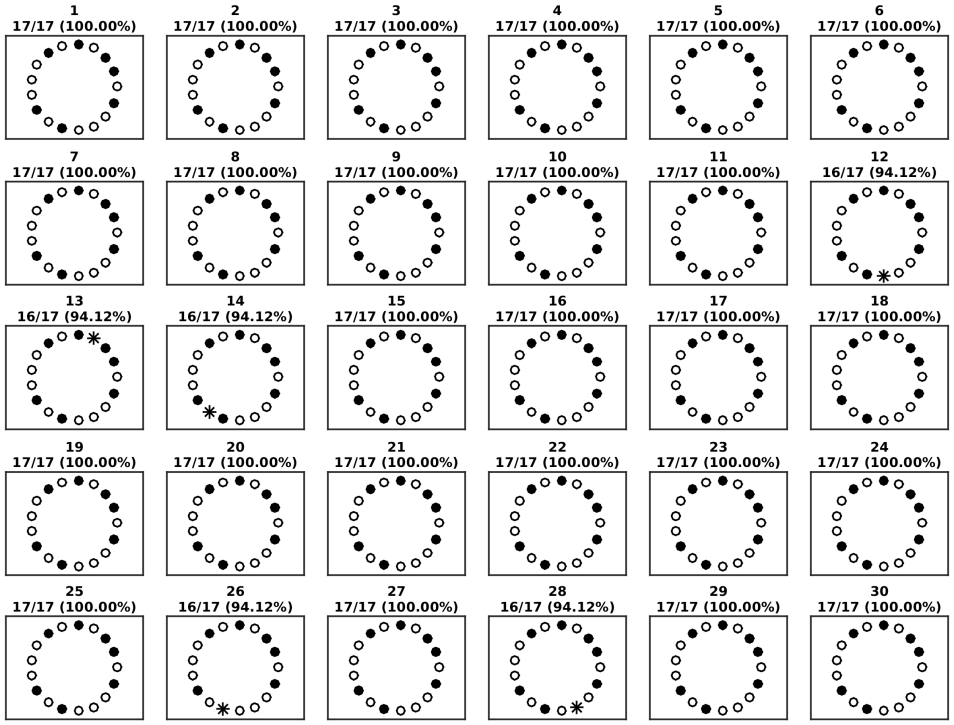
	Nombre de neurones cachés							
	0	1	2	3	4	5	8	10
Essais convergents (/30)	0	0	2	2	7	8	16	25
Taux moyen de classification (%)	86,3	76,9	87,1	90,2	93,3	94,3	97,3	99,0

Sur ce problème jugé difficile, l'algorithme et le MLQ offrent de bonnes performances pour un nombre suffisant de neurones cachés. En effet, même si le quantron seul échoue à la tâche, le MLQ 2-10-1 converge pour plus de 83 % des essais avec un taux de classification de 99,0 %.

La Figure 5.8a montre que même si l'algorithme n'arrive pas à entraîner entièrement le quantron seul, il l'amène néanmoins à classer correctement au moins 13 des 17 points pour toutes les tentatives. De plus, un seul pixel demeure mal classé dans huit d'entre elles (1, 6, 14, 16, 20, 22, 27 et 29), ce qu'on peut traduire comme la preuve d'une certaine efficacité de l'algorithme. Les pixels mal classés ne se situent pas dans une région particulière de l'heptadécagone, mais l'on remarque qu'au moins l'un des deux pixels éteints du coin inférieur gauche est classé incorrectement dans 21 des 30 essais. Il semble donc difficile de reproduire la séquence des points 8 à 16 avec succès (voir Figure 5.4f pour la numérotation des points).



(a) Quantron seul

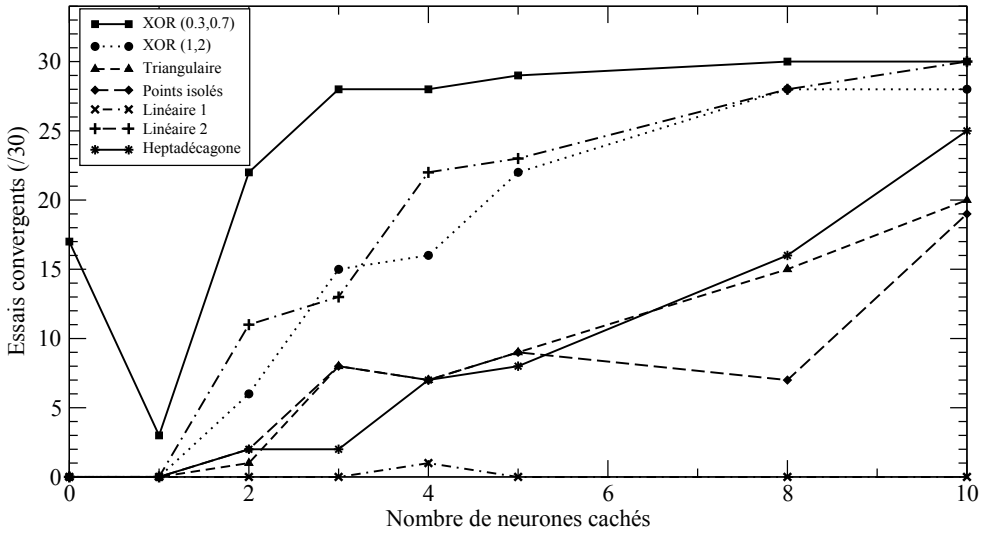


(b) MLQ 2-10-1

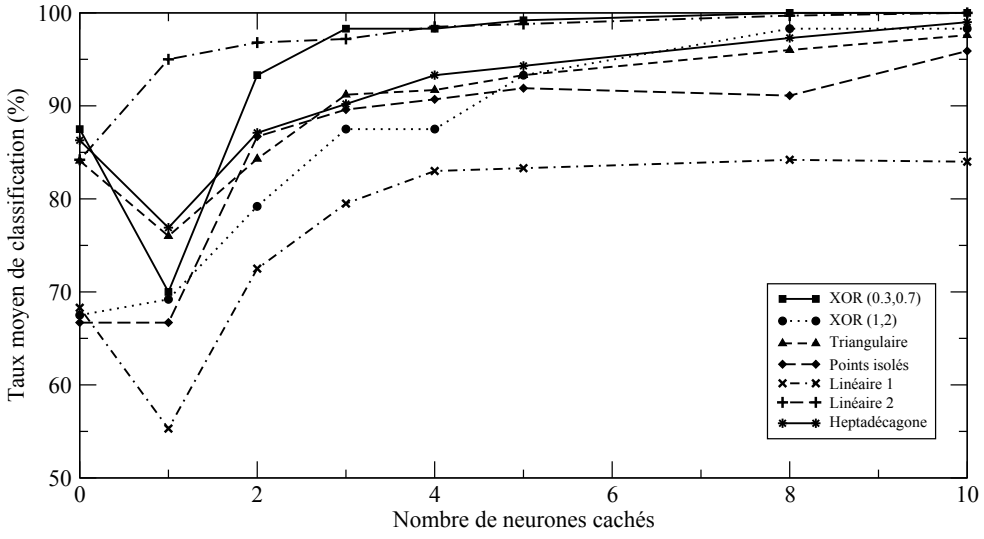
Figure 5.8 Mosaïques résultant de l'apprentissage sur l'heptadécagone

Même pour le MLQ 2-10-1 (Figure 5.8b), cette difficulté ne disparaît pas totalement, car on observe à la Figure 5.8b que l'unique pixel mal classé de quatre essais (12, 14, 26 et 28) parmi les cinq n'ayant pas convergé est localisé dans cette portion de l'image.

On clôt cette section en analysant la Figure 5.9b regroupant sous forme de graphes les performances de l'algorithme en fonction du nombre de neurones cachés pour tous les problèmes étudiés.



(a) Nombre d'essais convergents



(b) Taux moyen de classification correcte

Figure 5.9 Performances de l'algorithme en fonction de la taille de la couche cachée

Les graphiques confirment la tendance remarquée précédemment voulant que les performances s'améliorent plus on augmente la taille de la couche cachée. Ce comportement est tout à normal puisque plus un modèle possède de paramètres ajustables, plus il devient flexible et plus il peut s'adapter à une situation donnée.

Cependant, une diminution des performances (surtout du taux de classification) en passant d'aucun à un seul neurone caché est observée pour la majorité des problèmes. On l'explique par le fait que le quantron de sortie, qui détermine la classe du point, ne possède qu'une seule entrée (voir Figure 5.10). Cette entrée doit alors être nécessairement excitatrice (poids synaptique positif) pour que sa fonction d'activation $v(t)$ ait la possibilité d'atteindre le seuil qui est lui-même positif. Or, comme il a été discuté dans le cadre des deux problèmes linéairement séparables, un quantron seul agit comme une porte logique de seuil x_c. Si $x \leq x_c$, le quantron est actif tandis que si $x > x_c$, il demeure silencieux (voir Figure 5.7).

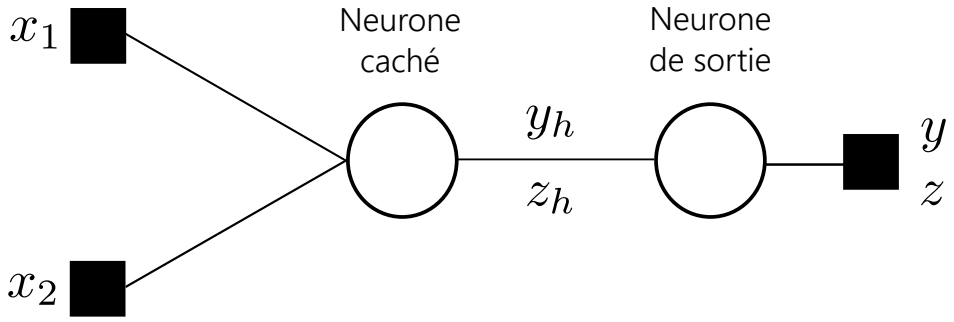


Figure 5.10 Schéma d'un MLQ 2-1-1 : le quantron de sortie possède une seule entrée.

Dans ce contexte, le neurone de sortie est actif seulement si le quantron caché est actif et qu'il génère une sortie $y_h \leq x_c$. Au contraire, le neurone de sortie est silencieux uniquement si le neurone caché est inactif ou s'il est actif et qu'il produit une sortie $y_h > x_c$. Pour que le MLQ résolve le problème, il faut alors que le neurone caché puisse transformer les coordonnées d'un point en une valeur qui soit inférieure ou supérieure à x_c selon la classe du point. Conséquemment, la tâche que doit accomplir l'unité cachée (et par conséquent le MLQ 2-1-1) devient généralement plus complexe que celle qui est demandée à un seul quantron (être actif pour une classe de pixels et rester inactif pour l'autre). Cette observation est vraisemblablement ce qui cause la diminution de performance observée pour la majorité des problèmes.

Mis à part ce cas particulier, le fait que les performances s'améliorent en ajoutant des neurones cachés démontre que l'algorithme est en mesure de gérer adéquatement un nombre élevé

de paramètres variables et d'exploiter la puissance qu'ils offrent pour reproduire de plus en plus fidèlement les formes cibles. Ainsi, la solution présentée permet d'entraîner des quantrons, individuellement ou en réseaux, sur une grande variété de problèmes de classification.

La discussion entamée dans ce chapitre se poursuit plus en détail au chapitre suivant dans lequel, entre autres, on examine les avantages et inconvénients de la solution proposée et comment elle se compare à l'algorithme SpikeProp.

CHAPITRE 6 DISCUSSION

Les deux premières sections de ce chapitre se concentrent respectivement sur l'analyse des profils de courbes d'erreur en lien avec la convergence de l'algorithme, et sur l'influence du paramètre d'approximation λ des sigmoïdes sur ses performances. Ensuite, on compare la nouvelle méthode avec SpikeProp sur les problèmes du XOR et de l'heptadécagone avant de clore par une discussion portant sur ses avantages et inconvénients.

6.1 Courbes d'erreur et convergence

Ayant constaté que la méthode produit de bons résultats, on s'intéresse maintenant aux courbes d'erreur et à la convergence. On débute en observant l'évolution de l'erreur de classification ($2E_z$) pour les caractères alphabétiques.

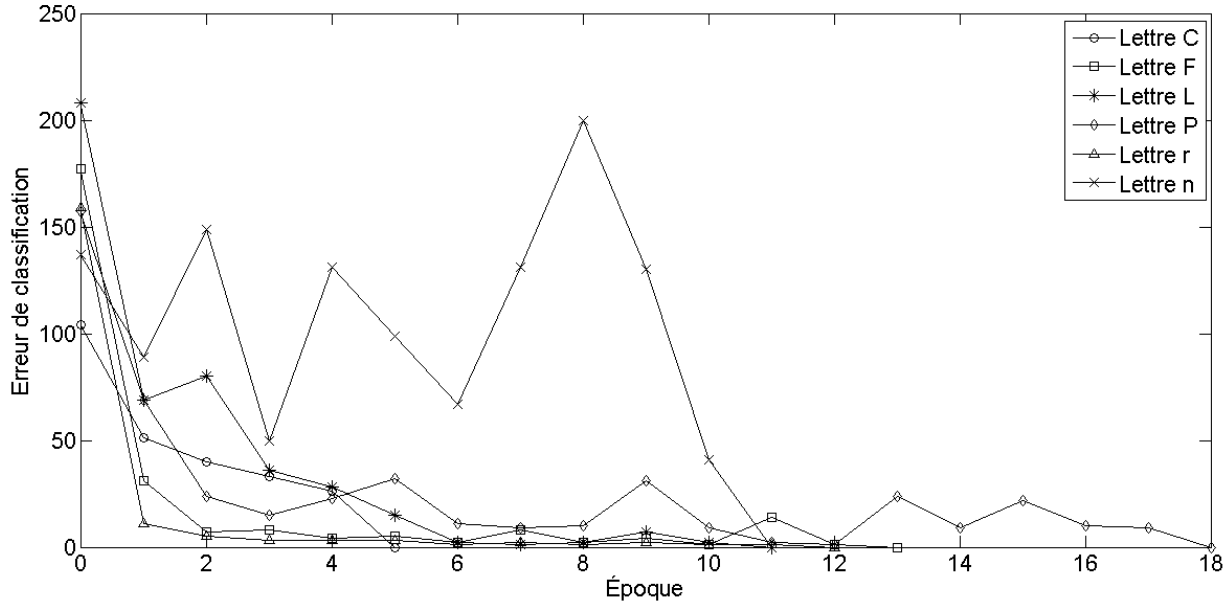
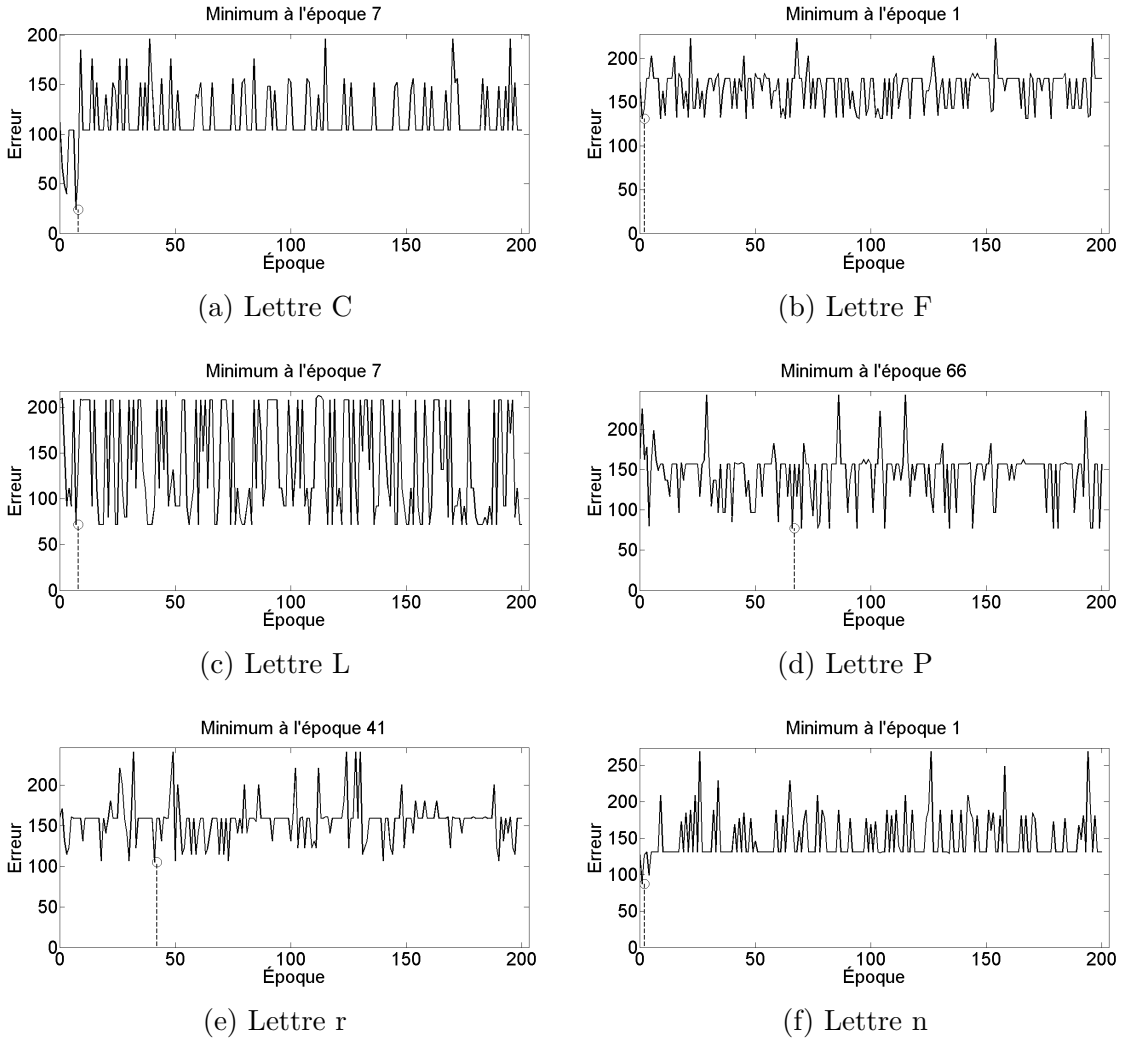


Figure 6.1 Courbes d'erreur pour les caractères alphabétiques (essais convergents)

La Figure 6.1 illustre des courbes d'erreur typiques pour des essais convergents. En général, l'erreur diminue fortement dès la première époque, puis continue de s'amoindrir à un rythme plus lent jusqu'à devenir nulle très rapidement, après moins de 20 époques pour les cas exposés. On remarque que l'erreur n'est pas décroissante. Elle a tendance à fluctuer lorsqu'elle est presque nulle, juste avant d'atteindre la convergence. Elle peut aussi croître de manière importante au cours de l'apprentissage, comme c'est le cas pour la lettre n aux époques 7 et 8, pour ensuite redescendre brusquement.

Le fait que les courbes soient non monotones reflète deux phénomènes. D'une part, la surface d'erreur s'avère rugueuse parce que la sortie du quantron est une fonction discontinue de ses paramètres. D'autre part, l'hypothèse SpikeProp et les dérivées symboliques utilisées par l'algorithme ne représentent pas la surface d'erreur exacte : il arrive donc que les modifications apportées provoquent une augmentation de l'erreur. Ces situations engendrent toutefois une bonne exploration de l'espace des paramètres, puisque certaines configurations sont visitées même si elles accroissent temporairement l'erreur. De ces positions à première vue défavorables, l'algorithme parvient à se rediriger ensuite vers des vallées de plus en plus creuses de la surface d'erreur jusqu'à converger.



(a) Lettre C

(b) Lettre F

(c) Lettre L

(d) Lettre P

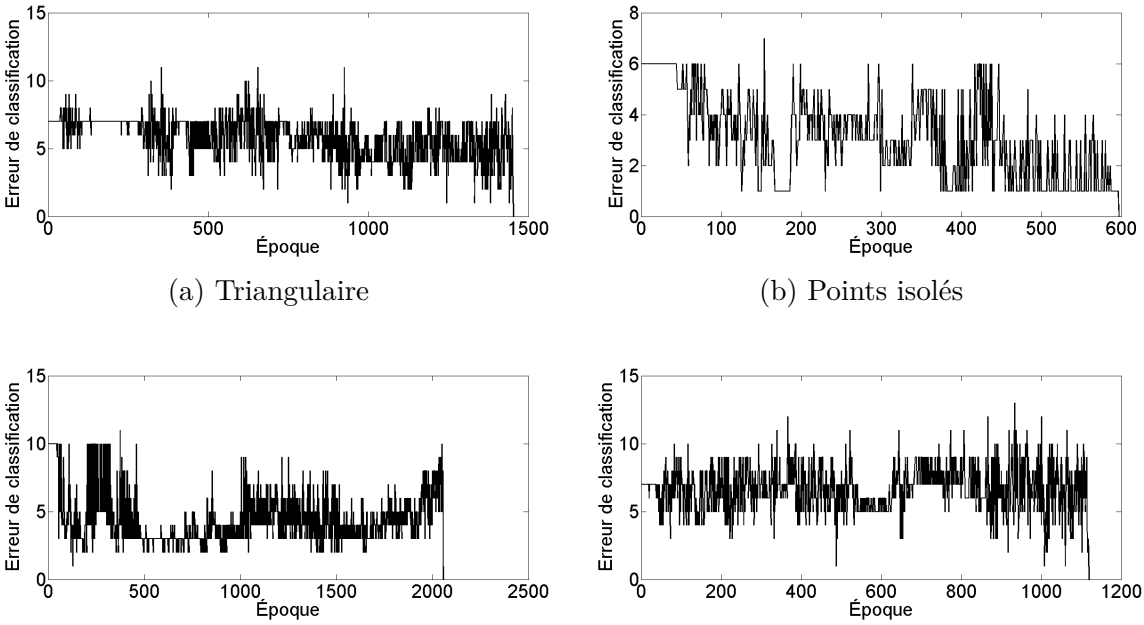
(e) Lettre r

(f) Lettre n

Figure 6.2 Courbes d'erreur pour les caractères alphabétiques (essais divergents)

Évidemment, ceci n'est pas le cas pour tous les essais et la Figure 6.2 présente l'évolution de courbes d'erreur dans des circonstances de divergence. On observe que généralement, un

minimum est atteint tôt dans la procédure d'apprentissage, puis l'erreur fluctue autour d'une valeur fixe. Dans tous les cas, cela survient lorsque le poids d'une entrée est augmenté près du seuil et que les autres poids sont fortement diminués, devenant même souvent négatifs. En voulant rendre actifs certains pixels inactifs, le poids dominant est augmenté et dépasse le seuil et vice versa, de sorte que l'erreur oscille. La valeur du plateau sous-jacent aux oscillations correspond aux époques où le poids dominant franchit le seuil, soit au nombre de pixels éteints sur l'image cible. L'algorithme n'arrive pas à s'extirper de ces situations où une entrée vient à dominer le maximum de la fonction d'activation. Cela peut s'expliquer par le fait que la dérivée de ce maximum est plus d'une fois approchée par la dérivée de la fonction d'activation au temps où elle l'atteint. Puisque pour la quasi-totalité des exemples, la même entrée devient majoritairement responsable de l'atteinte du maximum, ce sont ses paramètres qui sont le plus souvent modifiés au détriment des autres.



(a) Triangulaire

(b) Points isolés

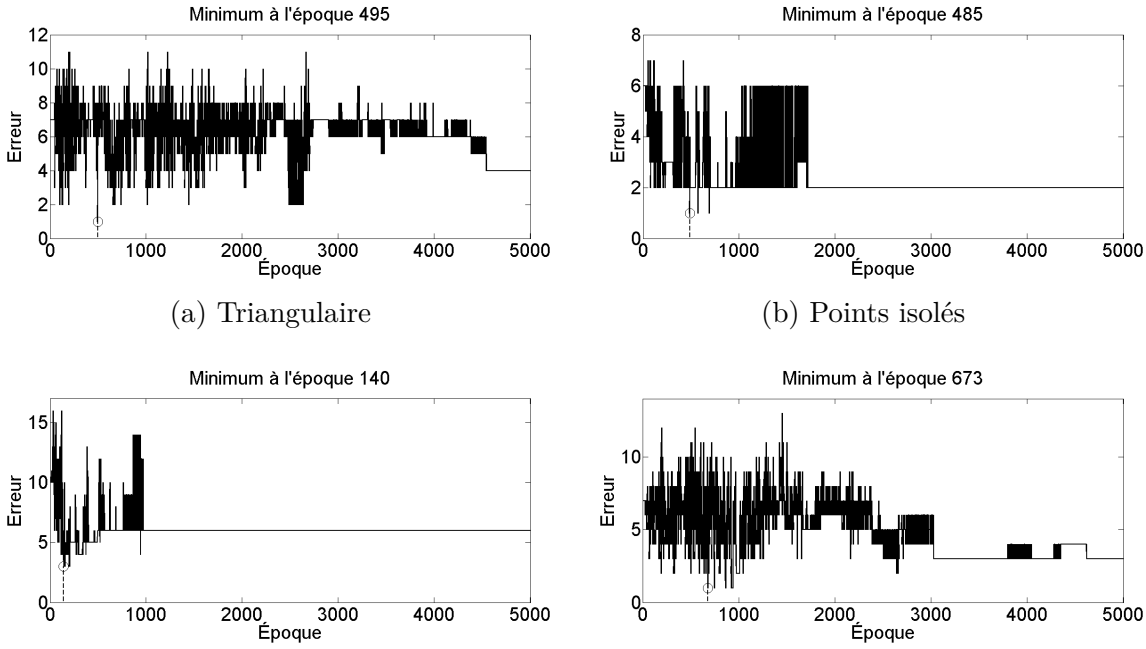
(c) Linéaire 2

(d) Heptadécagone

Figure 6.3 Courbes d'apprentissage avec les états d'activation (essais convergents)

On poursuit en examinant des courbes d'apprentissage typiques d'un MLQ 2-10-1 pour des essais ayant convergé. Contrairement à la Figure 6.1, la Figure 6.3 ne montre pas de diminution significative de l'erreur au cours des premières itérations. On ne remarque même pas de tendance générale à décroître continûment et graduellement vers une erreur nulle. On constate que l'erreur fluctue régulièrement et parfois de façon brusque, bien qu'il existe certaines périodes présentant une certaine stabilité. Pour les problèmes triangulaire, des points

isolés et de l'heptadécagone, la trajectoire de l'algorithme visite plus souvent des configurations de faible erreur avant de finalement converger. Par contre, comme le montre l'essai particulier du problème linéaire 2, il peut même arriver que l'erreur tende à s'accroître juste avant d'atteindre une erreur nulle. Cela démontre encore une fois à quel point la surface d'erreur est irrégulière.



(a) Triangulaire

(b) Points isolés

(c) Linéaire 1

(d) Heptadécagone

Figure 6.4 Courbes d'apprentissage avec les états d'activation (essais divergents)

Enfin, on s'intéresse aux courbes d'apprentissage divergentes de la Figure 6.4. Les comportements durant les portions initiales diffèrent très peu de ceux observés pour les essais convergents. Cependant, les courbes contrastent de par la présence de régions totalement stagnantes. Celles-ci apparaissent lorsque tous les neurones de la couche cachée deviennent silencieux pour tous les exemples mal classés. À ce moment, les règles d'apprentissage ne modifient plus aucun paramètre et le réseau tombe dans une configuration qu'il ne peut plus quitter. Il s'agit là d'une lacune importante de l'algorithme, car c'est la seule situation où les neurones silencieux bloquent le processus d'apprentissage. Dans ces circonstances, l'absence de convergence est ainsi due à une désensibilisation progressive de la couche cachée aux stimuli présentés.

6.2 Influence du paramètre d'approximation λ des sigmoïdes

Le paramètre d'approximation λ des sigmoïdes appraraît aux équations (4.19) à (4.21) et deux des trois heuristiques optimales l'utilisent. La Figure 6.5 illustre son influence sur la performance de la méthode pour les problèmes du XOR (1,2) et de l'heptadécagone. Il est clair que pour ces deux tâches, la valeur optimale parmi celles qui ont été testées est $\lambda = 1$. Celle-ci a d'ailleurs été employée dans toutes les expériences présentées précédemment. Lorsque λ s'écarte de cette valeur, on observe une baisse du nombre d'essais convergents et du taux moyen de classification spécialement marquée dans le cas du problème de l'heptadécagone.

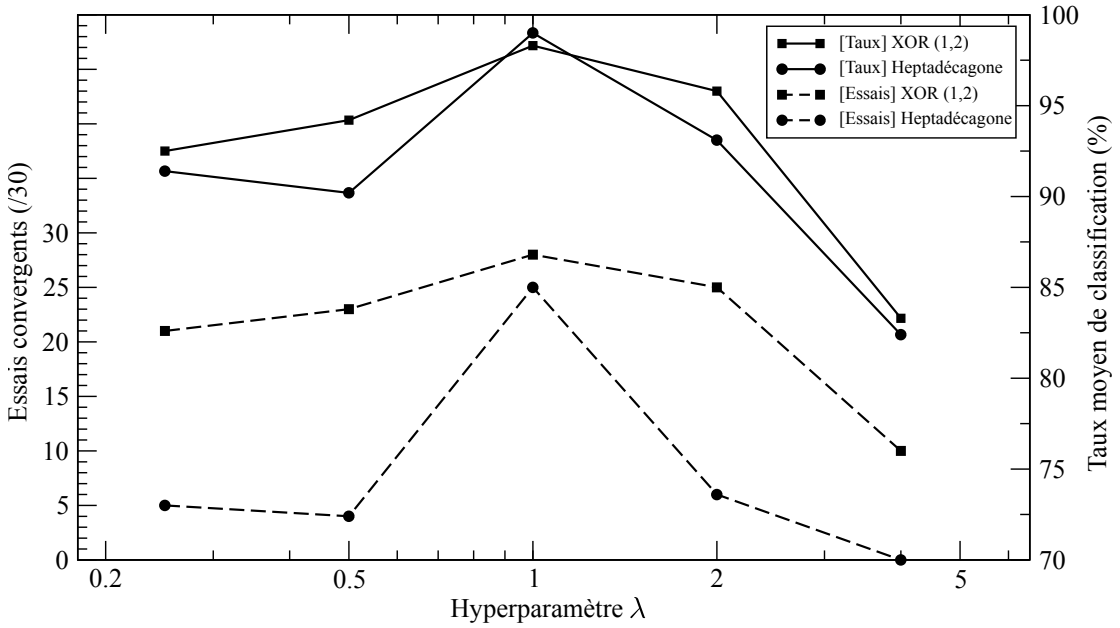


Figure 6.5 Influence du paramètre d'approximation λ des sigmoïdes sur la performance

L'existence d'une valeur intermédiaire optimale peut s'expliquer en analysant les effets de valeurs extrêmes. Si λ est grand, la magnitude de la dérivée de la sigmoïde est significative seulement lorsque le maximum de $v(t)$ est très proche du seuil ; autrement, elle s'avère à toutes fins pratiques négligeable et ne contribue pas aux changements de paramètres. Au contraire, si λ est petit, le profil de la dérivée de la sigmoïde centré sur Γ est plus étendu, mais sa magnitude maximale — égale à $\lambda/4$ — est également faible : sa contribution s'en voit ainsi encore une fois limitée. Bien qu'aucune méthode systématique n'a été développée pour identifier la valeur optimale de λ, la sensibilité des performances face à cette dernière suggère qu'elle doit être judicieusement choisie.

6.3 Comparaison des perfomances

Comme la solution proposée se base partiellement sur le formalisme de SpikeProp, il s'avère pertinent de comparer les performances des deux algorithmes. La comparaison se limite ici au problème du XOR étant donné qu'il s'agit du seul problème résolu par l'algorithme proposé dans ce mémoire et qui apparaît systématiquement dans la littérature entourant SpikeProp. Avant toute chose, on rappelle que SpikeProp travaille avec des entrées et une sortie qui correspondent aux instants des impulsions reçues et émises. Il convient donc d'expliquer brièvement comment SpikeProp parvient à résoudre le problème du XOR.

6.3.1 Problème du XOR temporel

Pour qu'un réseau SpikeProp puisse résoudre le XOR, on doit encoder le problème temporellement. Ainsi, on associe à chaque état binaire (0 et 1) une impulsion à un instant précis, en prenant soin d'assigner un temps plus grand à l'état 0 qu'à l'état 1. L'encodage généralement utilisé représente les états d'entrées par des impulsions à 0 ms (1) et 6 ms (0) et les états de sortie par des impulsions à 10 ms (1) et 16 ms (0) (Bohte *et al.*, 2002; Schrauwen et Campenhout, 2004; Booij et tat Nguyen, 2005; McKennoch *et al.*, 2006; Delshad *et al.*, 2010). Par conséquent, le réseau employé doit toujours posséder deux entrées et une sortie. Si les entrées sont identiques (0-0 ms ou 6-6 ms), alors le neurone de sortie doit émettre une impulsion à 16 ms (0). Si les entrées sont distinctes (0-6 ms ou 6-0 ms) alors ce même neurone doit produire une impulsion à 10 ms (1). Ce comportement attendu est résumé au Tableau 6.1, où le symbole $\oplus$ représente la fonction logique OU exclusif.

Tableau 6.1 Encodage temporel du problème du XOR

Entrées (ms)	Sortie (ms)	Équivalence logique
(0,0)	16	$1 \oplus 1 = 0$
(0,6)	10	$1 \oplus 0 = 1$
(6,0)	10	$0 \oplus 1 = 1$
(6,6)	16	$0 \oplus 0 = 0$

L'architecture la plus souvent employée est un réseau 3-5-1 à cinq neurones cachés. La présence d'un troisième neurone d'entrée qui émet toujours au temps de référence (à 0 ms) est essentielle. En effet, un réseau à deux entrées ne pourrait produire la même sortie pour les entrées (0,0) et (6,6) puisque la seconde représente une situation identique à la première, à un décalage temporel près. Si ce réseau produit une impulsion à $t = t_1$ pour l'entrée (0,0), alors il génère nécessairement un impulsion à $t = t_1 + 6$ pour l'entrée (6,6), ce qui le rend inapte à résoudre le XOR tel que défini. La troisième entrée fixée à 0 ms s'avère donc indispensable

pour un réseau SpikeProp (Sporea et Gruning, 2011). Cette caractéristique n'est pas requise pour un réseau de quantrons, car les états binaires ne sont pas encodés temporellement.

6.3.2 Performances sur le problème du XOR

Le Tableau 6.2 repertorie les performances de l'algorithme SpikeProp original et de plusieurs de ses variantes sur le problème du XOR décrit précédemment. Il rassemble également les résultats des travaux de Lastère (2005) et de de Montigny (2014).
D'abord, SpikeProp a été appliqué de nombreuses fois sur le problème du XOR temporel avec un réseau 3-5-1 comportant 320 poids. L'article original de Bohte *et al.* (2002) affirme que pour un choix optimal du taux d'apprentissage, la totalité des 10 essais ont convergé. Des simulations indépendantes du même problème menées par Booij et tat Nguyen (2005) et Delshad *et al.* (2010) ont produit des taux de convergence supérieurs à 90 % sur 100 essais. En bornant supérieurement la norme de l'ajustement Δw permis à chaque itération, Booij et tat Nguyen (2005) sont parvenus à un taux de convergence de 100 % pour un nombre d'essais identique. Dans leurs travaux, Delshad *et al.* (2010) ont tenté d'améliorer la convergence par diverses techniques intégrant un taux d'apprentissage et un terme de *momentum* adaptatifs (méthodes DS η-α, mod-DS η et mod-DS η-α) sans toutefois dépasser un taux de convergence de 78 %. Enfin, SpikeProp étendu entraîne les poids, les délais et les largeurs de noyaux, ce qui lui permet d'obtenir un taux de convergence de 90 % avec un réseau 3-5-1 ayant près de trois fois moins de paramètres. En voulant réduire la taille du réseau à trois neurones cachés, le taux chute toutefois à 60 %.

Parmi les travaux antérieurs sur le quantron, la méthode de Lastère (2005) a convergé pour 75 des 100 essais pour un quantron seul entraîné sur le problème du XOR(3,5). Avec la rétropropagation lisse, de Montigny (2014) parvient à un taux de 90 % sur le XOR(1,2) avec un seul quantron encore une fois. Par contre, il utilise un paramètre de plus (le seuil Γ est entraîné) et la performance est évaluée sur 10 fois moins d'essais, ce qui n'est pas négligeable. Lorsque qu'un réseau 2-5-1 à 51 paramètres est employé, le taux obtenu est de 100 %, toujours calculé sur 10 essais.

La solution proposée dans ce mémoire permet de résoudre le problème du XOR avec beaucoup moins de paramètres que les réseaux SpikeProp. En effet, avec un réseau 2-10-1 à 90 paramètres (comparativement à 320 paramètres), des taux supérieurs à 93 % sont atteints sur les deux problèmes du XOR considérés. On constate donc que le nouvel algorithme s'avère plus efficient que SpikeProp et ses variantes pour résoudre ce problème de référence.

Tableau 6.2 Performances de plusieurs algorithmes sur le problème du XOR

Algorithme		Réseau	Nombre de paramètres	Meilleur taux de convergence (%) avec nombre d'essais
SpikeProp	Original (Bohte *et al.*, 2002)	3-5-1	320	100 (10)
	Original (Booij et tat Nguyen, 2005)			96 (100)
	Original (Delshad *et al.*, 2010)			91 (100)
	$\|\Delta w\|$ borné (Booij et tat Nguyen, 2005)			100 (100)
	DS η-α (Delshad *et al.*, 2010)			78 (100)
	mod-DS η (Delshad *et al.*, 2010)			76 (100)
	mod-DS η-α (Delshad *et al.*, 2010)			78 (100)
	Étendu (Schrauwen et Campenhout, 2004)	3-5-1	126	90
		3-3-1	76	60
XOR(3,5) Lastère (2005)		2-0-1	6	75 (100)
Rétropropagation lisse XOR(1,2) de Montigny (2014)		2-0-1	7	90 (10)
		2-5-1	51	100 (10)
Solution proposée	XOR(0,3 ;0,7)	2-0-1	6	57 (30)
		2-5-1	45	97 (30)
		2-10-1	90	100 (30)
	XOR(1,2)	2-0-1	6	0 (30)
		2-5-1	45	73 (30)
		2-10-1	90	93 (30)

Les solutions de Lastère (2005) et de de Montigny (2014) montrent également cette efficience accrue. On peut comparer la solution proposée à celle de Lastère (2005) sur les problèmes du XOR où la coordonnée la plus grande n'est pas un multiple entier de la plus petite (XOR(3,5) et XOR(0,3 ;0,7), voir Section 5.2). Pour un seul quantron, on associe à l'algorithme de Lastère (2005) un taux de convergence supérieur (75 % contre 57 %). Par contre, il faut rappeler que ce résultat est issu d'une approximation de la fonction d'activation $v(t)$ du quantron par une

série de Fourier tronquée au deuxième ordre, elle-même appliquée à un noyau rectangulaire assez différent du noyau original. On peut alors expliquer l'apparente baisse de performance de la nouvelle solution proposée ici par la difficulté inhérente à entraîner le quantron avec un noyau ressemblant davantage au noyau original et en utilisant l'expression exacte de $v(t)$. En ajoutant des neurones cachés, cette solution n'a aucune difficulté à résoudre le XOR(0,3 ;0,7).

On peut également comparer le nouvel algorithme à la rétropropagation lisse sur le problème du XOR(1,2). À la fois pour le quantron seul et pour le réseau 2-5-1, la solution proposée produit des taux de convergence inférieurs. En fait, cette dernière doit utiliser 10 neurones cachés pour atteindre un taux semblable à ceux obtenus par de Montigny (2014). Cette observation peut s'expliquer de plusieurs façons. D'abord, la rétropagation lisse permet d'entraîner les seuils Γ de chaque quantron, donnant une puissance supplémentaire au modèle. Ensuite, le critère de convergence a été spécifié en fonction de l'évaluation lisse de la sortie du réseau, et non en fonction de l'évaluation exacte. Le résultat présenté au Tableau 6.2 correspond à l'évaluation exacte, mais les résultats détaillés de l'article montrent qu'il est fréquent que la performance de la méthode sur l'évaluation exacte soit inférieure. Cet aspect doit être considéré dans la comparaison des deux méthodes et, bien que la solution proposée ici ne présente pas de meilleurs taux de convergence, elle demeure efficace et il serait sans doute hâtif de conclure qu'elle est inférieure sans effectuer au préalable une analyse plus poussée.

La nature récente de l'algorithme SpikeProp et des travaux dont il a fait l'objet a pour conséquence qu'il ne semble pas exister, au moment d'écrire ces lignes, de logiciel implémentant cette méthode, qu'il soit libre (*open source*) ou propriétaire. Dans le cadre de travaux futurs, il serait ainsi pertinent d'appliquer SpikeProp sur d'autres problèmes que le XOR, tels que le problème de l'heptadécagone de la Section 5.2 ou celui de la classification des pixels de représentations de caractères alphabétiques afin d'approfondir davantage cette comparaison.

Enfin, certaines méthodes statistiques pourraient être intégrées à cette analyse comparative dans le futur, à savoir la méthode de classification par les k plus proches voisins (k-NN) ou encore certaines méthodes de régression pouvant être étendues à des tâches de classification telles que la régression logistique (voir Bishop (2006)) ou la *projection pursuit regression* (Friedman et Stuetzle, 1981). Ceci étant dit, il est important de se rappeler que l'objectif principal visé par ce travail était d'abord et avant tout de développer un algorithme d'apprentissage pour le quantron parce que ce dernier n'en possède pas. Dans la mesure où, à l'avenir, la solution proposée pourra graduellement être appliquée à des problèmes de complexité croissante (reconnaissance de caractères, d'images ou de vidéos, par exemple), il deviendra alors

certainement intéressant de considérer ces méthodes statistiques dans l'analyse comparative.

6.4 Avantages et désavantages de la méthode proposée

Les analyses précédentes ont permis de mettre en évidence deux lacunes importantes de l'algorithme proposé. On élabore ici sur ces dernières et on suggère des avenues possibles pour tenter de les corriger, avant de discuter des avantages de la méthode.

Un premier aspect de la méthode qui l'amène à diverger est l'approximation utilisée pour la dérivée du maximum de la fonction d'activation par rapport aux paramètres du quantron. Puisque celle-ci consiste à évaluer la dérivée de la fonction d'activation seulement en $t = \xi$, cela suppose que tous les autres points de $v(t)$ ne l'influencent pas. L'avantage de cette approximation est qu'elle est rapide à évaluer, mais elle peut s'avérer inexacte dans certaines situations où, par exemple, un changement de paramètres donné mène à une diminution de $v(t)$ autour de son maximum et à une augmentation ailleurs de sorte que son maximum global ne décroît pas en réalité. Pour corriger cette lacune, il faudrait trouver une façon d'exprimer la dérivée en fonction de la forme de la fonction d'activation et non par sa valeur en un point. Avec (1.9), de Montigny (2014) a proposé d'utiliser une moyenne généralisée des valeurs de $v(t)$ prises sur un intervalle considéré. Quand le paramètre d'approximation λ tend vers l'infini, l'approximation revient toutefois à dériver $v(t)$ évaluée au temps où elle est maximale. Ainsi, plus elle est exacte, plus elle s'approche de l'approximation employée par le présent algorithme. Son désavantage principal réside dans sa lourdeur d'implémentation, puisque l'intégrale doit être évaluée en partitionnant l'intervalle avec les temps de l'ensemble T auxquels la dérivée d'un PPS triangulaire varie. Néanmoins, il serait intéressant d'intégrer une approximation de la sorte au présent algorithme afin de mesurer l'impact sur ses performances.

Le deuxième désavantage est l'incapacité de la méthode à poursuivre l'apprentissage des paramètres si tous les neurones de la couche cachée deviennent silencieux pour un exemple donné. Lorsque cette situation surgit, la sortie du quantron de sortie vaut alors $y_j = \xi_j = 0$ étant donné que $z_h = 0$ pour toutes ses entrées. Par conséquence, les dérivées de l'état d'activation

$$\frac{\partial z_j}{\partial z_h}, \quad \frac{\partial z_j}{\partial y_h}, \quad \frac{\partial z_j}{\partial w_{hj}}, \quad \frac{\partial z_j}{\partial \theta_{hj}}, \quad \frac{\partial z_j}{\partial s_{hj}}$$

s'annulent. D'abord, parce que les dérivées par rapport aux paramètres P_{hj} sont nulles, les paramètres du neurone de sortie restent inchangés selon (4.15). Ensuite, les gradients locaux $\delta_{zy,h}$ et $\delta_{zz,h}$ de la couche cachée s'annulent aussi selon (4.23) et (4.24) parce que les dérivées

de z_j par rapport à z_h et à y_h sont égales à zéro. Il en résulte que les paramètres des neurones cachés stagnent en vertu de (4.25).

Il est tout à fait logique que les paramètres du neurone de sortie ne soient pas modifiés si toutes ses entrées sont silencieuses, puisque ces changements n'auront aucun effet sur sa sortie tant et aussi longtemps qu'au moins un neurone caché ne s'active et contribue à $v_j(t)$. La faille se situe donc dans la définition de la dérivée de z_j par rapport à z_h qui s'annule parce que $\xi_j = 0$. Une avenue possible pour remédier au phénomène de stagnation pourrait être de redéfinir ξ_j dans le cas spécifique où $z_h = 0 \ \forall h$ par le maximum (ou le minimum selon la situation) qui serait atteint par $v_j(t)$ si toutes les entrées avaient été actives. Sinon, une heuristique pourrait être créée qui donnerait une valeur positive à $\frac{\partial z_j}{\partial z_h}$ si $w_{hj} > 0$ et une valeur négative si $w_{hj} < 0$. Elle exprimerait le fait que si le neurone h s'active, il tend à activer le neurone de sortie s'il produit des PPS excitateurs et à le désactiver s'il génère des PPS inhibiteurs. On pourrait aussi envisager un mécanisme qui détecterait certaines circonstances menant potentiellement à une couche cachée silencieuse (par exemple, lorsque les poids synaptiques deviennent faibles ou encore négatifs) afin de rediriger l'algorithme vers des configurations plus favorables.

L'algorithme présente tout de même plusieurs avantages. En premier lieu, il permet d'entraîner le quantron en utilisant son formalisme exact, c'est-à-dire en utilisant les expressions exactes des trains de PPS et de la fonction d'activation après avoir choisi un noyau plus simple se substituant au noyau original. Ensuite, il s'applique autant à un quantron seul qu'à un MLQ, ce qui n'est pas le cas de tous les algorithmes développés par le passé. Comme il a été observé, cela permet d'exploiter le potentiel du MLQ en reconnaissance de formes et de résoudre certains problèmes complexes nécessitant une couche cachée pour projeter les entrées dans un espace de dimension supérieure. L'algorithme reste simple à implémenter et s'avère assez rapide grâce aux propriétés des PPS triangulaires qui simplifient grandement l'évaluation de $v(t)$. Finalement, un dernier avantage évident est qu'il offre de bonnes performances sur la grande majorité des problèmes étudiés pour autant que l'on munisse le MLQ d'un nombre suffisant de neurones cachés.

CHAPITRE 7 CONCLUSION

On conclut en récapitulant d'abord le travail accompli présenté dans ce mémoire. On revient par la suite sur les limites de la solution proposée, avant de suggérer certaines avenues de recherches futures visant à l'améliorer ou à l'appliquer à de nouveaux problèmes.

7.1 Synthèse des travaux

L'objectif principal des travaux, qui était de concevoir un algorithme d'apprentissage supervisé pour entraîner un réseau de quantrons multicouche à résoudre des problèmes de classification, a été accompli avec succès. L'idée de départ consistait à exploiter un lien apparent entre les formalismes du quantron et des neurones à impulsions. En explorant la littérature à leur sujet, il a été remarqué qu'une partie importante du mécanisme du neurone SpikeProp, basé sur le *Spike Response Model* plus général, est partagée avec le quantron. Plus spécifiquement, les deux modèles possèdent une fonction d'activation dépendante du temps définissant la sortie du neurone via le premier temps où elle franchit un seuil. Ainsi, le Théorème 1 établit une équivalence entre les paramètres des deux types de neurones, ouvrant du même coup la porte à une adaptation de l'algorithme SpikeProp au quantron.

Afin de préparer le terrain pour le développement du nouvel algorithme, le noyau triangulaire — utilisé en substitution au noyau original — a été étudié en détail. Les valeurs de ses dérivées premières partielles par rapport à t et à s ont été fixées et le Lemme 2 a permis entre autres d'accélérer significativement l'évaluation des sorties du quantron en restreignant la recherche de l'extremum de la fonction d'activation à un ensemble fini de temps. Le dernier résultat important concernant le noyau triangulaire est la dérivation d'une expression analytique pour la valeur de son extremum en fonction des paramètres x, w, s et N. Celle-ci sert à proposer des heuristiques particulières pour que l'algorithme puisse gérer les neurones silencieux et apprendre en utilisant les états d'activation, ce que SpikeProp n'est pas en mesure de faire.

Bien que l'objectif premier soit de permettre à un MLQ de reproduire des états d'activation cibles z_d, la solution proposée se veut plus générale, car elle couvre aussi l'apprentissage des sorties analogiques y_d lorsque la cible est active. L'algorithme est ainsi développé en partant du principe de la rétropropagation de l'erreur et en définissant quatre gradients locaux. L'hypothèse SpikeProp est utilisée pour estimer la variation de la sortie y d'un quantron produite par un changement de ses paramètres. Pour la portion de l'algorithme ayant trait aux états d'activation et aux neurones silencieux, la variable intermédiaire ξ est définie. Elle

représente l'instant où le maximum (s'il est strictement positif) ou le minimum (autrement) de la fonction d'activation est atteint, et sert à redéfinir la sortie y lorsque le quantron est silencieux. De plus, z est introduit dans l'expression de la fonction d'activation afin que l'état d'activation apparaisse dans les dérivées en chaîne de la BP. Rappelons que cela ne constitue pas une approximation puisque $z \in \{0, 1\}$. Suite à ces modifications, les dérivées de z par rapport aux deux sorties des neurones en amont ont été définies symboliquement. Pour ce faire, on fait comme si z était approché par une sigmoïde dont l'argument est la différence entre le maximum de $v(t)$ et le seuil. En posant de plus que la dérivée de ce maximum est égale à la dérivée de $v(t)$ évaluée en ξ, on dérive l'approximation analytique de z pour définir les dérivées symboliques figurant dans les expressions des gradients locaux.

À ce stade, il restait à définir trois dernières dérivées symboliques, à savoir celles de l'état d'activation par rapport à chaque classe de paramètres w, θ et s. À cet effet, cinq heuristiques ont été proposées pour chacune d'elles dont certaines se basent sur les rôles particuliers joués par les paramètres. Afin de déterminer quelle combinaison d'heuristiques était optimale, l'algorithme a été appliqué sur un quantron devant reproduire des images binaires de caractères alphabétiques générées par un quantron. Les deux sorties cibles y_d et z_d étaient alors disponibles pour l'entraînement. Chaque heuristique a été testée individuellement et les performances moyennes sur les six formes ont servi à retenir les deux meilleures pour chaque type de paramètres. Les huit agencements possibles ont finalement été testés et la combinaison H 3-3-4 est ressortie gagnante. Les taux de classification correcte moyen et minimum atteints, 98,5 % et 94,6 % respectivement, s'avèrent très satisfaisants et confirment que l'algorithme fonctionne correctement. Avec ces résultats encourageants, l'algorithme a été testé sur les mêmes images en utilisant seulement les états d'activation cibles z_d. Sur des quantrons seuls, une baisse de performance a été observée. Par contre, en utilisant des réseaux à cinq et dix neurones cachés, la méthode est parvenue à retrouver son efficacité en affichant des taux moyens de classification supérieurs à 90 %.

Une série de problèmes jouets a aussi été étudiée pour mesurer les performances de la méthode pour ce mode d'apprentissage spécifique (z_d uniquement) : le XOR, les frontières triangulaire et linéaires, les points isolés et l'heptadécagone. Il a été observé que l'algorithme livre de bonnes performances sur la majorité des problèmes pour autant qu'il dispose de suffisamment de neurones cachés. Seul le premier problème de frontière linéaire déroge à cette tendance. Pour celui-ci, un seul essai a convergé : il s'agit d'un essai avec un MLQ 2-4-1. Cela prouve que le problème est résoluble, mais qu'il est ardu de repérer une solution avec la méthode avancée. Sachant que les performances se sont avérées largement supérieures pour

le deuxième problème linéaire où les classes des points sont inversées, on comprend qu'un problème et son complémentaire peuvent présenter des niveaux de complexité bien différents concernant l'apprentissage. Sur les autres problèmes, le taux moyen de classification correcte surpasse les 95 % pour un MLQ 2-10-1, ce qui démontre que l'algorithme est en mesure de gérer adéquatement un grand nombre de neurones pour résoudre ces tâches avec une grande efficacité.

L'analyse des courbes d'erreur a permis de révéler deux désavantages majeurs de la procédure d'apprentissage proposée. Premièrement, il arrive que l'algorithme se dirige vers une configuration où le poids d'une entrée devient dominant et les autres entrées sont réprimées. Puisque les modifications sont fondées pour la plupart sur les valeurs des dérivées de la fonction d'activation à son maximum, elles affectent uniquement l'entrée dominante à toutes fins pratiques et la trajectoire n'arrive plus à s'extirper de ces configurations défavorables. Deuxièmement, bien que la méthode parvient à bien gérer les neurones silencieux en général, il existe une situation particulière qui l'entrave à un tel point que l'apprentissage s'arrête. Si tous les neurones de la couche cachée sont silencieux en même temps pour un exemple donné, les dérivées symboliques de l'état d'activation du neurone de sortie s'annulent et empêchent toute modification non seulement aux paramètres du neurone de sortie, mais aussi à ceux des neurones cachés. Si cela survient pour tous les exemples incorrectement classés au cours de l'apprentissage, la procédure stagne totalement et diverge par le fait même. L'élimination de ces deux lacunes permettrait sans doute de réduire le nombre d'essais divergents.

7.2 Avenues de recherche future

Malgré les résultats satisfaisants obtenus, il reste beaucoup de travail à faire pour bonifier la solution proposée et pour tester son potentiel.

D'abord, il serait intéressant de tenter de corriger les deux désavantages mentionnés précédemment. Il faudrait alors incorporer des expressions utilisant la forme globale de la fonction d'activation, du moins les valeurs des dérivées en plus d'un point, pour améliorer la qualité de l'assignation du crédit aux différentes entrées et éviter d'accorder trop de pouvoir à une seule entrée. De plus, une heuristique pourrait être développée pour empêcher la stagnation de l'apprentissage lorsque tous les neurones de la couche cachée deviennent silencieux.

Ensuite, il serait utile de réfléchir à une procédure d'initialisation des paramètres plus rigoureuse. Actuellement, des lois uniformes sont utilisées sur des intervalles qui se sont avérés empiriquement satisfaisants. S'il était possible de démarrer le réseau dans une configuration

minimisant la probabilité de divergence, les performances s'en verraient sûrement augmentées.

Lorsque l'algorithme a été appliqué aux images générées par un quantron, les sorties y_d étaient connues et évidemment cohérentes avec les états d'activation cibles z_d, ce qui a aidé grandement au processus d'apprentissage. Comme il a été mentionné, si l'on utilise seulement z_d, la méthode n'arrive plus à converger. Cela amène à se poser la question suivante : si l'algorithme ne parvient pas à reproduire une forme, serait-il possible de générer un ensemble de sorties y_d cohérentes qui pourrait faciliter l'apprentissage ? Il s'agit là d'un problème excessivement complexe qui n'a pu être résolu dans ce mémoire malgré le fait qu'il ait fait l'objet d'une réflexion certaine : pourrait-on, par exemple, ajuster les valeurs de y_d au fur et à mesure que l'apprentissage progresse de façon à obtenir un tel ensemble cohérent ? Quoi qu'il en soit, que cette question trouve oui ou non une réponse affirmative n'affecte pas la pertinence d'avoir intégré y_d dans la méthode. En effet, cette sortie pourrait être utilisée dans d'autres types d'encodage où les classes seraient par exemple représentées par le temps auquel émet le quantron de sortie et non pas par son état d'activation. L'algorithme travaillerait alors en mode régression dans l'optique de résoudre un problème de classification.

Finalement, bien qu'il ait été utilisé avec un seul quantron de sortie, le formalisme de la méthode peut supporter plusieurs neurones de sortie. On pourrait donc tester la procédure sur des problèmes multiclasses tels que le problème IRIS afin d'explorer davantage son potentiel et son efficacité. Également, la méthode demeure applicable si le réseau de quantrons possède plusieurs couches cachées, ce qui permettrait d'aborder ces problèmes avec des architectures profondes.

RÉFÉRENCES

Abramowitz, M. et Stegun, I. A. (1964). *Handbook of mathematical functions : with formulas, graphs, and mathematical tables*. No. 55. Courier Corporation.

Ahmed, F. Y. H., Shamsuddin, S. M. et Hashim, S. Z. M. (2013). Improved SpikeProp for using particle swarm optimization. *Mathematical Problems in Engineering*, *2013*, 1–13.

Barber, D. (2003). Learning in Spiking Neural Assemblies. *Advances in Neural Information Processing Systems 15 - Proceedings of the 2002 Conference, NIPS 2002*. 165–172.

Belatreche, A., Maguire, L. P., Mcginnity, M. et Wu, Q. X. (2003). An Evolutionary Strategy for Supervised Training of Biologically Plausible Neural Networks. *IEEE Cybernetics Intelligence Challenges and Advances (CICA) 2003*. 39–44.

Bi, G.-q. et Poo, M.-m. (1998). Synaptic Modifications in Cultured Hippocampal Neurons : Dependence on Spike Timing , Synaptic Strength , and Postsynaptic Cell Type. *The Journal of Neuroscience*, *18*(24), 10464–10472.

Bishop, C. M. (2006). *Pattern recognition and machine learning*, vol. 4. springer New York.

Bohte, S. M., Kok, J. N. et La Poutré, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, *48*(1-4), 17–37.

Booij, O. et tat Nguyen, H. (2005). A gradient descent rule for spiking neurons emitting multiple spikes. *Information Processing Letters*, *95*(6 SPEC. ISS.), 552–558.

Carnell, A. et Richardson, D. (2005). Linear Algebra for Time Series of Spikes. *Proceedings of ESANN*. 363–368.

Connolly, J.-F. et Labib, R. (2009). A multiscale scheme for approximating the Quantron's discriminating function. *IEEE transactions on neural networks*, *20*(8), 1254–66.

Covey, E. et Casseday, J. H. (1999). Timing in the auditory system of the bat. *Annual review of physiology*, *61*, 457–476.

de Montigny, S. (2007). *Étude sur l'apprentissage d'une approximation du quantron*. Mémoire de maîtrise, École Polytechnique de Montréal.

de Montigny, S. (2014). New approximation method for smooth error backpropagation in a quantron network. *Neural Networks*, *60*(0), 84 – 95.

de Montigny, S. et Labib, R. (2011). Learning algorithms for a specific configuration of the quantron. *The 2011 International Joint Conference on Neural Networks*. IEEE, 567–572.

Delshad, E., Moallem, P. et Monadjemi, S. (2010). Spiking neural network learning algorithms : Using learning rates adaptation of gradient and momentum steps. *Telecommunications (IST), 2010 5th International Symposium on*. 944–949.

Dermietzel, R. et Spray, D. C. (2013). Gap junctions, electric synapses. D. Pfaff, éditeur, *Neuroscience in the 21st Century*, Springer New York. 439–473.

Fang, H., Luo, J. et Wang, F. (2012). Fast learning in spiking neural networks by learning rate adaptation. *Chinese Journal of Chemical Engineering*, *20*(6), 1219–1224.

Fetz, E. E. et Gustafsson, B. (1983). Relation between shapes of post-synaptic potentials and changes in firing probability of cat motoneurones. *The Journal of physiology*, *341*, 387–410.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, *7*(2), 179–188.

FitzHugh, R. (1961). Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Biophysical Journal*, *1*(6), 445–466.

Freeth, T. (2009). Decoding an ancient computer. *Scientific American*, *301*(6), 76 – 83.

Friedman, J. H. et Stuetzle, W. (1981). Projection pursuit regression. *Journal of the American statistical Association*, *76*(376), 817–823.

Gautrais, J. et Thorpe, S. (1998). Rate coding versus temporal order coding : a theoretical approach. *Biosystems*, *48*(1–3), 57 – 65.

Gerstner, W. (1995). Time structure of the activity in neural network models. *Physical Review E*, *51*(1), 738–758.

Gerstner, W. et Kistler, W. M. (2002). *Spiking neuron models : Single neurons, populations, plasticity*. Cambridge university press.

Ghosh-Dastidar, S. et Adeli, H. (2009). A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural Networks*, *22*(10), 1419–1431.

Grüning, A. et Sporea, I. (2012). Supervised learning of logical operations in layered spiking neural networks with spike train encoding. *Neural Processing Letters*, *36*(2), 117–134.

Gulledge, A. T., Kampa, B. M. et Stuart, G. J. (2005). Synaptic integration in dendritic trees.

Gütig, R. et Sompolinsky, H. (2006). The tempotron : a neuron that learns spike timing-based decisions. *Nature neuroscience*, *9*(3), 420–428.

Hackenbeck-Lambert, J. (2011). *Étude du potentiel de création de formes du quantron*. Mémoire de maîtrise, École Polytechnique de Montréal.

Hamel, E. (2013). *Modélisation mathématique de la dépression synaptique et des périodes réfractaires pour le quantron*. Mémoire de maîtrise, École Polytechnique de Montréal.

Hamm, T. M., Sasaki, S., Stuart, D. G., Windhorst, U. et Yuan, C. S. (1987). Distribution of single-axon recurrent inhibitory post-synaptic potentials in a single spinal motor nucleus in the cat. *The Journal of physiology*, *388*, 653–664.

Haykin, S. (1999). *Neural Networks : A Comprehensive Foundation*. Prentice-Hall, seconde édition.

Hong, S., Ning, L., Xiaoping, L. et Qian, W. (2010). A cooperative method for supervised learning in Spiking neural networks. *Proceedings of the 2010 14th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2010*. Ieee, 22–26.

Hornik, K., Stinchcombe, M. et White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366.

Hsu, F. (2002). *Behind Deep Blue : Building the Computer that Defeated the World Chess Champion*. Princeton University Press.

Iansek, R. et Redman, S. J. (1973). The amplitude, time course and charge of unitary excitatory post-synaptic potentials evoked in spinal motoneurone dendrites. *The Journal of physiology*, *234*(3), 665–688.

Izhikevich, E. M. (2003). Simple model of spiking neurons.

Jack, J. J., Miller, S., Porter, R. et Redman, S. J. (1971). The time course of minimal excitory post-synaptic potentials evoked in spinal motoneurones by group Ia afferent fibres. *The Journal of physiology*, *215*(2), 353–380.

Labelle, J. et Mercier, A. (1993). *Introduction à l'analyse réelle*. Modulo.

Labib, R. (1999). New single neuron structure for solving nonlinear problems. *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, *1*, 617–620.

Labib, R. et De Montigny, S. (2012). On the learning potential of the approximated quantron. *International journal of neural systems*, *22*(3), 1250010.

Lapicque, L. (1907). Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *Journal de physiologie et de pathologie générale*, *9*(1), 620–635.

Lastère, R. (2005). *Conception de l'algorithme d'apprentissage supervisé d'un réseau multicouche de quantrons*. Mémoire de maîtrise, École Polytechnique de Montréal.

Legenstein, R., Naeger, C. et Maass, W. (2005). What can a neuron learn with spike-timing-dependent plasticity ? *Neural computation*, *17*(11), 2337–2382.

L'Espérance, P.-Y. (2010). *Modélisation de la transmission synaptique d'un neurone biologique à l'aide de processus stochastiques*. Mémoire de maîtrise, École Polytechnique de Montréal.

Lin, J. W. et Faber, D. S. (2002). Modulation of synaptic delay during synaptic plasticity.

Maass, W. (1997). Networks of spiking neurons : The third generation of neural network models. *Neural Networks*, *10*(9), 1659–1671.

Maass, W. et Bishop, C. M. (2001). *Pulsed neural networks*. MIT press.

Maass, W. et Schmitt, M. (1999). On the complexity of learning for spiking neurons with temporal coding. *Information Comput*, *153*(1), 26–46.

Magee, J. C. (2000). Dendritic integration of excitatory synaptic input. *Nature reviews. Neuroscience*, *1*(3), 181–190.

Masaru, F., Haruhiko, T., Hidehiko, K. et Terumine, H. (2008). Shape of error surfaces in spikeprop. *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. 840–844.

McCulloch, W. S. et Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

McKennoch, S., Liu, D. L. D. et Bushnell, L. (2006). Fast Modifications of the SpikeProp Algorithm. *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 3970–3977.

Minsky, M. L. et Papert, S. (1988). Perceptrons : an introduction to computational geometry (expanded edition).

Moore, S. C. (2002). *Back-propagation in spiking neural networks*. Thèse de doctorat, University of Bath.

Panzeri, S., Petersen, R. S., Schultz, S. R., Lebedev, M. et Diamond, M. E. (2001). The role of spike timing in the coding of stimulus location in rat somatosensory cortex. *Neuron*, *29*(3), 769–777.

Paugam-Moisy, H. et Bohte, S. (2012). Computing with Spiking Neuron Networks. G. Rozenberg, T. Bäck et J. N. Kok, éditeurs, *Handbook of Natural Computing*, Springer Berlin Heidelberg, Berlin, Heidelberg. 335–376.

Pavlidis, N. G., Tasoulis, D. K., Plagianakos, V. P., Nikiforidis, G. et Vrahatis, M. N. (2005). Spiking neural network training using evolutionary algorithms. *Proceedings of the International Joint Conference on Neural Networks*. IEEE, vol. 4, 2190–2194.

Pepga Bissou, J. (2007). *Conception d'un algorithme d'apprentissage pour un réseau de quantrons*. Thèse de doctorat, École Polytechnique de Montréal.

Pfister, J.-P., Barber, D. et Gerstner, W. (2003). Optimal Hebbian Learning : A Probabilistic Point of View. *Artificial Neural Networks and Neural Information Processing — ICANN/ICONIP 2003*, *2714*, 92–98.

Pfister, J.-P., Toyoizumi, T., Barber, D. et Gerstner, W. (2006). Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural computation*, *18*(6), 1318–1348.

Pillow, J. W., Paninski, L. et Simoncelli, E. P. (2004). Maximum likelihood estimation of a stochastic integrate-and-fire neural encoding model. *Neural computation*, *16*(12), 2533–2561.

Ponulak, F. (2005). ReSuMe - New Supervised Learning Method for Spiking Neural Networks. Rapport technique, Poznan University of Technology.

Ponulak, F. et Kasiński, A. (2010). Supervised learning in spiking neural networks with ReSuMe : sequence learning, classification, and spike shifting.

Rosenblatt, F. (1958). The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Rumelhart, D. E., Hinton, G. E. et Williams, R. J. (1986). Learning representations by back-propagating errors.

Sayer, R. J., Friedlander, M. J. et Redman, S. J. (1990). The time course and amplitude of EPSPs evoked at synapses between pairs of CA3/CA1 neurons in the hippocampal slice. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, *10*(3), 826–836.

Schrauwen, B. et Campenhout, J. V. (2004). Extending SpikeProp. *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, *1*, 471–475.

Senn, W., Schneider, M. et Ruf, B. (2002). Activity-dependent development of axonal and dendritic delays, or, why synaptic transmission should be unreliable. *Neural computation*, *14*(3), 583–619.

Shrestha, S. B. et Song, Q. (2015). Adaptive learning rate of spikeprop based on weight convergence analysis. *Neural Networks*, *63*, 185 – 198.

Sporea, I. et Gruning, A. (2011). Reference time in SpikeProp. *The 2011 International Joint Conference on Neural Networks*, 1090–1092.

Strain, T. J., McDaid, L. J., McGinnity, T. M., Maguire, L. P. et Sayers, H. M. (2010). An STDP training algorithm for a spiking neural network with dynamic threshold neurons. *International journal of neural systems*, *20*(6), 463–480.

Taherkhani, A., Belatreche, A., Li, Y. et Maguire, L. (2015). Dl-resume : A delay learning-based remote supervised method for spiking neurons. *Neural Networks and Learning Systems, IEEE Transactions on*, *PP*(99), 1–1.

Takase, H., Fujita, M., Kawanaka, H., Tsuruoka, S., Kita, H. et Hayashi, T. (2009). Obstacle to training SpikeProp networks - Cause of surges in training process. *Proceedings of the International Joint Conference on Neural Networks*. 3062–3066.

Thiruvarudchelvan, V., Crane, J. W. et Bossomaier, T. (2013). Analysis of SpikeProp convergence with alternative spike response functions. *Proceedings of the 2013 IEEE Symposium on Foundations of Computational Intelligence, FOCI 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013*. Ieee, 98–105.

Thorpe, S., Delorme, A. et Van Rullen, R. (2001). Spike-based strategies for rapid processing. *Neural Networks*, *14*(6-7), 715–725.

Thorpe, S. J. et Imbert, M. (1989). Biological constraints on connectionist modelling. *Connectionism in perspective*, 1–36.

Tiňo, P. et Mills, A. (2005). Learning beyond finite memory in recurrent networks of spiking neurons. L. Wang, K. Chen et Y. Ong, éditeurs, *Advances in Natural Computation*, Springer Berlin Heidelberg, vol. 3611 de *Lecture Notes in Computer Science*. 666–675.

VanRullen, R., Guyonneau, R. et Thorpe, S. J. (2005). Spike times make sense.

Vázquez, R. A. et Garro, B. A. (2011). Training spiking neurons by means of particle swarm optimization. *Advances in Swarm Intelligence*, Springer. 242–249.

Wickens, A. (2009). *Introduction to Biopsychology*. Pearson Education.

Williams, S. R. et Stuart, G. J. (2000). Site independence of EPSP time course is mediated by dendritic I(h) in neocortical pyramidal neurons. *Journal of neurophysiology*, *83*(5), 3177–3182.

Wu, Q. X., McGinnity, T. M., Maguire, L. P., Glackin, B. et Belatreche, A. (2006). Learning under weight constraints in networks of temporal encoding spiking neurons. *Neurocomputing*, *69*(16-18), 1912–1922.

Yan, L., Haruhiko, T., Hiroharu, K. et Shinji, T. (2012). Improve discontinuous output in SpikeProp — Effective type of weight decay. *The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems*, 1915–1918.

Yang, J., Yang, W. et Wu, W. (2012). A remark on the error-backpropagation learning algorithm for spiking neural networks. *Applied Mathematics Letters*, *25*(8), 1118–1120.

ANNEXE A DÉRIVÉES PREMIÈRES DU NOYAU ORIGINAL DU QUANTRON

On développe ici les expressions analytiques du noyau original $\varepsilon_0(t;s)$, décrit à l'équation (1.2), utilisées pour la comparaison avec le noyau triangulaire de la Section 3.1.

Pour obtenir les expressions des dérivées premières, il faut d'abord exprimer la fonction $Q(z)$ sous sa forme intégrale. On obtient

$$Q(z) = \frac{1}{\sqrt{2\pi}} \int_z^\infty \exp\left(-\frac{x^2}{2}\right) dx \tag{A.1}$$

D'après la règle de Leinbniz de dérivation sous le signe de l'intégration (Abramowitz et Stegun, 1964), si

$$F(x) = \int_{a(x)}^{b(x)} f(x,y)dy,$$

alors

$$\frac{dF(x)}{dx} = f(x,a(x))\frac{d}{dx}a(x) - f(x,b(x))\frac{d}{dx}b(x) + \int_{a(x)}^{b(x)} \frac{\partial f(x,y)}{\partial y} dy.$$

La dérivée de $Q(z)$ par rapport à z vaut alors

$$\frac{dQ(z)}{dz} = -\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)\Big|_{x=z} \frac{dz}{dz} = -\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) \tag{A.2}$$

Dérivée première par rapport à t

La dérivée du noyau original par rapport au temps s'obtient directement à partir de (A.2).

$$\varepsilon_0'(t;s) = \begin{cases} -\frac{\ln a}{2\sqrt{2\pi}Q(\frac{\ln a}{\sqrt{s}})} t^{-3/2} \exp\left(-\frac{\ln^2 a}{2t}\right) & \text{si } 0 \le t < s \\ \frac{\ln a}{2\sqrt{2\pi}Q(\frac{\ln a}{\sqrt{s}})} (t-s)^{-3/2} \exp\left(-\frac{\ln^2 a}{2(t-s)}\right) & \text{si } s < t < 2s \\ 0 & \text{autrement} \end{cases} \tag{A.3}$$

On peut montrer en utilisant la règle de L'Hospital que la dérivée est continue en $t = 0$ et qu'elle y est nulle. Par contre, la dérivée est discontinue en $t = s$ et en $t = 2s$. Puisque les expressions obtenues dans cette annexe servent uniquement à visualiser le comportement des dérivées, ces questions ne sont pas abordées plus en détail. Remarquons simplement que la dérivée par rapport au temps est négative sur $[0, s[$ et positive sur $]s, 2s[$.

Dérivée première par rapport à s

La dérivée par rapport à s est plus complexe à évaluer, car la constante de normalisation du noyau dépend de s. Les expressions sur l'intervalle $]0, s[$ et sur $]s, 2s[$ diffèrent donc significativement.

Sur $]0, s[$, on trouve

$$\frac{\partial \varepsilon_0(t;s)}{\partial s} = -\frac{\ln a}{2\sqrt{2\pi}} \frac{Q(\frac{\ln a}{\sqrt{t}})}{Q^2(\frac{\ln a}{\sqrt{s}})} s^{-3/2} \exp\left(-\frac{\ln^2 a}{2s}\right), \qquad 0 < t < s \tag{A.4}$$

tandis qu'il faut appliquer la règle de dérivation d'un produit pour obtenir l'expression sur $]s, 2s[$.

$$\frac{\partial \varepsilon_0(t;s)}{\partial s} = \frac{\ln a}{2\sqrt{2\pi} Q(\frac{\ln a}{\sqrt{s}})} \left[\frac{Q(\frac{\ln a}{\sqrt{t-s}})}{Q(\frac{\ln a}{\sqrt{s}})} s^{-3/2} \exp\left(-\frac{\ln^2 a}{2s}\right) + (t-s)^{-3/2} \exp\left(-\frac{\ln^2 a}{2\,(t-s)}\right) \right], \qquad s < t < 2s \tag{A.5}$$

La dérivée première par rapport à s est également négative sur $]0, s[$ et positive sur $]s, 2s[$.

ANNEXE B ENSEMBLES SOLUTIONS DES IMAGES DE CARACTÈRES ALPHABÉTIQUES

Les valeurs des paramètres w, θ et s utilisées pour générer les représentations de caractères alphabétiques de la Section 5.1 avec un seul quantron sont répertoriées au Tableau B.1. Les ensembles des lettres C, F, L et P proviennent de Hackenbeck-Lambert (2011). Dans tous les cas, le seuil est fixé à $\Gamma = 1$ et chaque train est composé de $N = 10$ PPS triangulaires.

Tableau B.1 Valeurs des paramètres utilisées pour produire les caractères alphabétiques

Lettre	Entrée	w	θ	s
C	x_1	0,3644	0,0000	0,8141
	x_2	0,9468	7,6765	0,4069
F	x_1	0,2502	0,0000	0,8352
	x_2	0,6723	4,6413	0,2782
L	x_1	0,7329	0,0000	0,0415
	x_2	0,2872	6,1019	1,0342
r	x_1	0,3507	6,2055	0,7908
	x_2	0,4102	0,0000	0,8988
n	x_1	0,9006	6,2212	0,2576
	x_2	0,8805	0,0000	0,3199
P	x_1	0,5486	7,6452	0,3484
	x_1	0,1829	9,6786	1,3074
	x_2	0,3126	0,0000	1,0922